Market-Based Runtime Resource Allocation for LLM Multi-Agent Systems

Yixue Huang
School of Electronic Information
Wuhan University
Wuhan 430072, China
2023182120066@whu.edu.cn

Mingxin Wang
School of Electronic Information
Wuhan University
Wuhan 430072, China
wangmingxin@whu.edu.cn

Hao Li
School of Electronic Information
Wuhan University
Wuhan 430072, China
whulh@whu.edu.cn

Hao Jiang*
School of Electronic Information
Wuhan University
Wuhan 430072, China
jh@whu.edu.cn

Abstract

LLM multi-agent systems rely on division of labor to solve complex tasks, but their execution can be blocked by contention for shared resources such as GPU test environments, CPU execution slots, and exclusive tool sessions. Existing workflows, task allocation methods, and runtime schedulers usually arbitrate such access indirectly through workflow position, dependency structure, queue state, or resource metrics. These criteria do not fully capture the value of satisfying a specific request, because request value is distributed across agents' local execution states and changes with tests, retries, and downstream results. We propose a market-based protocol for access arbitration in LLM multi-agent systems. Each agent converts its current plan and local execution state into a structured, budget-constrained bid that signals relative urgency. For asynchronous requests, an online clearing method based on optimal stopping decides when to close the clearing window, allocate access rights, settle payments, and track resource occupation until release. Across three contention scenarios, Market reduces makespan by 5.2%, 2.5%, and 4.9% relative to the strongest primary baselines, while improving task score by about two percentage points.

1 Introduction

The advantage of LLM multi-agent systems lies in division of labor and collaboration. Different agents can take planning, coding, testing, checking, and repair roles to advance complex tasks. During execution, however, collaboration can be affected by contention for shared execution resources. In software development, multiple agents may request the same GPU test environment to validate code changes. In data analysis, multiple agents may request CPU execution slots or exclusive tool sessions to run scripts. When several agents request the same limited resource, serving one agent can make others wait, causing task blocking and overall delay. LLM multi-agent systems therefore need access arbitration for shared execution resources: the system must decide the access order, the waiting arrangement, and the timing of resource release under exclusive occupation.

The key to this arbitration is to determine the value of each request to the overall task, show in fig 1. Orchestrators and runtime scheduling modules maintain the task workflow, the request queue, and

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

the resource occupation state. These centralized records support resource access decisions, but they do not fully contain request value. That value is also distributed across the local execution states of individual agents. For example, one GPU test may validate a change on the critical path and unblock downstream work, while another may perform a routine check. The two requests may have the same resource type and expected occupation duration, but different effects on task progress. Because request value can change with later tests, retries, and downstream results, the information needed for arbitration cannot be fully recovered from centralized records alone.

Existing LLM multi-agent systems usually arbitrate shared resources only indirectly. Collaboration frameworks organize agent actions through roles, dialogue protocols, and workflow control [28, 8, 6, 19, 3, 15]. Task-decomposition methods assign subtasks or dependencies to agents, so request importance is often inherited from the task structure [2, 14, 9, 26]. Runtime and serving systems schedule GPU, CPU, and memory access using visible system signals such as arrival time, expected duration, resource capacity, queue state, memory cost, or fairness counters [29, 17, 12, 18, 24, 20]. These criteria can determine the access order, but they do not directly capture the marginal value of satisfying a specific request for overall task progress. This gap motivates request-level access arbitration that incorporates task value into shared execution resource allocation.

To bring task value at the request level into resource arbitration, agents need to express local resource demands in a form that is comparable, constrained, and translatable into access decisions. Hayek's analysis of distributed knowledge states that key knowledge in complex systems is distributed across participants' local circumstances and is difficult for a central mechanism to collect and compute completely in real time [7]. Effective coordination requires local judgments to be compressed into allocation signals. In scarce resource allocation, market mechanisms provide such signals: bids make distributed demands comparable, while budgets and payments make demand expression carry cost. Inspired by this idea, this paper introduces a market mechanism into access arbitration for shared execution resources in LLM multi-agent systems, so that agents can express the task value of resource requests through comparable signals that carry cost.

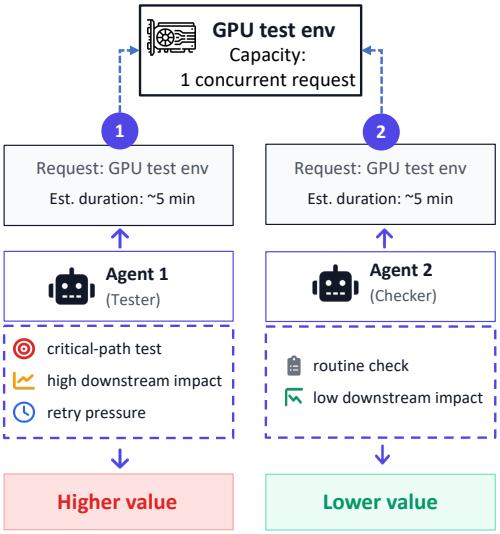


Figure 1: Same request, different task value. Two agents request the same GPU test environment with similar expected duration, but their local task states imply different values for task progress.

Applying a market mechanism to access arbitration for shared execution resources in LLM multi-agent systems requires two adaptations. First, the value of a resource request is not a prespecified valuation. It is formed during execution by the agent's current plan, local execution state, downstream dependencies, retry pressure, and expected occupation duration. The mechanism needs to transform this local information into comparable demand signals. Second, agents' resource requests arrive asynchronously. If the first request is cleared immediately after arrival, the mechanism may miss a higher value request that arrives later. If it waits too long, it may create new task blocking. The mechanism therefore needs to decide when to clear under asynchronous request arrivals.

Based on this idea, this paper proposes a protocol with budget constraints for signaling task value and arbitrating access. At the protocol layer, an agent generates a structured bid according to its current plan and local execution state. The bid represents the relative urgency of the current resource request and is constrained by the remaining budget. The budget constraint links the current request to future competitive capacity, so stronger demand expression has an intertemporal cost. For asynchronously arriving bids, this paper designs an online clearing method based on optimal stopping. After the first request arrives, the mechanism opens a candidate clearing window and balances waiting for more requests against preventing the current request from being further blocked. When continued waiting is unlikely to bring sufficient improvement in clearing quality, the window closes, and the

bids within the window are cleared together. After clearing, the mechanism allocates access rights, settles payments, and tracks resource occupation until release.

This paper makes three contributions.

1. We propose a budget-constrained protocol for signaling task value and arbitrating access, where agents express local resource demand via structured, budget-limited bids (§4.2).
2. We design an online clearing method based on optimal stopping that, under asynchronous bid arrivals, adaptively selects a clearing window, allocates access, settles payments, tracks occupation, and clears again after release (§4.3).
3. We provide theoretical analysis and controlled experiments. Across three contention scenarios, Market reduces makespan by 5.2%, 2.5%, and 4.9% relative to the strongest primary baselines, while modestly improving task score. Matched S3 comparisons further demonstrate lower allocator decision time and communication overhead relative to centralized value-aware schedulers (§5.4).

2 Related Work

Existing work relevant to our problem falls into two lines. First, LLM multiagent systems coordinate agents through collaboration, planning, and runtime serving. AutoGen [28], MetaGPT [8], AgentScope [6], ChatDev [19], AgentVerse [3], and CAMEL [15] organize roles, conversations, and workflows. Planning methods such as Amayuelas et al. [2], Li et al. [14], OWL [9], and TDAG [26] assign subtasks to agents. Runtime systems such as Justitia [29], Autellix [17], Nalar [12], AIOS [18], Llumnix [24], and Sheng et al. [20] schedule GPU, CPU, memory, or request queues. These works provide implicit access arbitration through workflow order, task structure, or runtime records, but they do not directly elicit the task value of each resource request from the requesting agent. Second, market mechanisms use bids, prices, and budgets to coordinate allocation under dispersed information [7]. Prior systems such as Spawn [25], market oriented programming [27], Tycoon [11], and fixed budget bidding [5] show that local demand can guide compute allocation. Detailed in Appendix A

3 Preliminary

3.1 Runtime Resource Allocation Setting

We consider an LLM multi-agent system with agents $\mathcal{A} = \{a_1, \ldots, a_N\}$ and shared runtime resources $\mathcal{R} = \{r_1, \ldots, r_M\}$. Each resource $r \in \mathcal{R}$ has capacity $c_r \in \mathbb{N}_+$. Access to one unit of capacity is exclusive during occupation. The object allocated by the mechanism is therefore a temporary access right, not ownership of the resource.

During execution, an agent may reach a step that requires a shared resource. At time t, agent a_i then generates a request $q_{i,t} = (r_{i,t}, d_{i,t})$, where $r_{i,t} \in \mathcal{R}$ is the requested resource and $d_{i,t} \in \mathbb{R}_+$ is the expected occupation duration. After demand elicitation, the submitted action is $x_{i,t} = (r_{i,t}, d_{i,t}, \beta_{i,t})$, where $\beta_{i,t} \in \mathbb{R}_{\geq 0}$ is the bid. Each agent has a remaining budget $B_{i,t} \in \mathbb{R}_{\geq 0}$, and every valid bid must satisfy $0 \leq \beta_{i,t} \leq B_{i,t}$. All agents start with the same initial budget $B_{i,0} = B_0$. Budgets are internal credits of the mechanism and are not LLM token budgets.

3.2 Information Structure and Objective

Each request has a latent *one-step value* $v_{i,t} \in \mathbb{R}_+$. It is the marginal benefit to agent a_i from receiving access in the current decision epoch rather than waiting for a later epoch. This value captures the expected reduction in critical-path delay or downstream blocking loss, net of the opportunity cost induced by occupying the resource for duration $d_{i,t}$. The value $v_{i,t}$ is private to the agent and is not directly observed by the allocator. The bid $\beta_{i,t}$ is the only demand signal exposed to the mechanism.

At time t, the runtime state is

$$S_t = (\mathcal{O}_t, \mathcal{Q}_t, \mathbf{B}_t, \mathcal{H}_t),$$

where $\mathcal{O}_t$ is the set of active occupation records, $\mathcal{Q}_t$ is the pending request set, $\mathbf{B}_t = (B_{1,t}, \ldots, B_{N,t})$ is the budget vector, and $\mathcal{H}_t$ contains public resource-level information such as

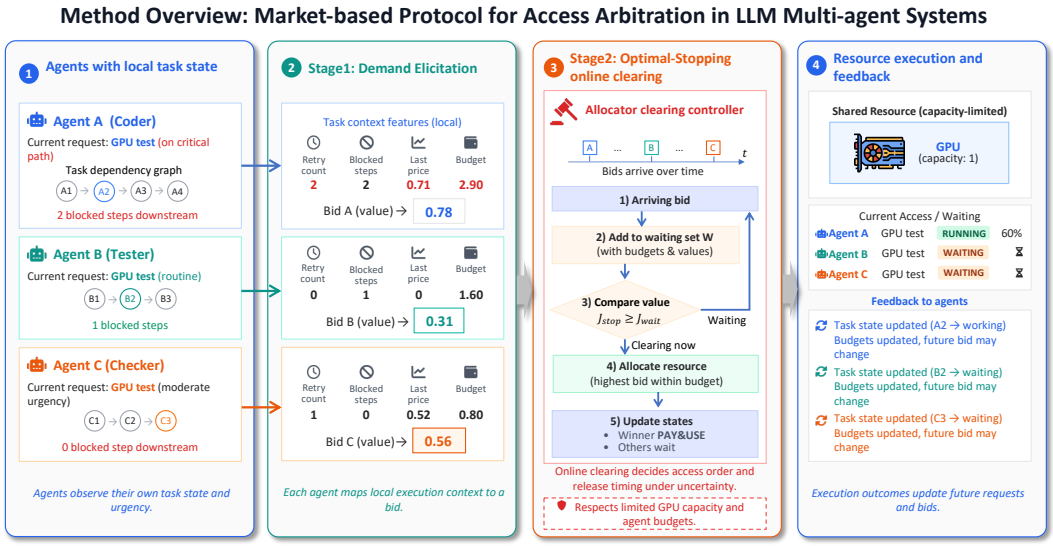


Figure 2: Overview of the proposed market-based access arbitration protocol. Each agent maps its local task state to a budget-constrained bid. The clearing controller accumulates asynchronously arriving bids, applies an optimal-stopping rule to decide when to clear, allocates the limited resource through first-price clearing, and feeds execution and budget updates back to agents.

queue lengths and recent clearing prices. An offline allocator with access to all latent values would solve

$$\max_{\pi} \sum_{t} \sum_{i \in W_t^{\pi}} v_{i,t}, \qquad W_t^{\pi} \text{ feasible under capacity and occupation constraints} \tag{1}$$

where π is an allocation policy and W_t^{π} is the set of agents granted access at time t. This objective is not directly attainable online because the allocator observes requests, bids, budgets, and occupation states, but not $v_{i,t}$.

3.3 Auction Primitives and Scope

A clearing event is created when multiple valid requests contend for the same resource. For resource r in clearing round e, let $\mathcal{P}_e^r \subseteq \mathcal{Q}_t$ denote the eligible request set, and let $\tilde{c}_r$ denote the currently allocable capacity. An allocation rule selects a winner set $W_e \subseteq \mathcal{P}_e^r$ with $|W_e| \leq \tilde{c}_r$. A payment rule assigns each participant a payment $p_i^e \geq 0$, with non-winners paying zero. If $|\mathcal{P}_e^r| \leq \tilde{c}_r$, any work-conserving rule can grant all eligible requests; ranking matters only in contested rounds.

4 Method

4.1 Mechanism Overview

Figure 2 summarizes the proposed two-stage market mechanism. Stage 1 (§4.2) maps each agent's local task state to a structured, budget-feasible bid. Stage 2 (§4.3) resolves asynchronous contention through an optimal-stopping-based online clearing rule, followed by first-price settlement and occupation tracking. Clearing outcomes update budgets that in turn constrain subsequent elicitation. The orchestration layer continues to manage task decomposition, role assignment, and workflow ordering independently of the mechanism.

4.2 Stage 1: Demand Elicitation

For the same access opportunity, the latent one-step value $v_{i,t}$ may differ across agents depending on their subtask phase, downstream blocking, expected occupation duration, and remaining budget.

The mechanism therefore does not ask agents to report a cardinal valuation. It asks each agent to produce a budget-feasible relative demand signal generated from local task context.

When agent a_i reaches a resource-dependent execution node at time t, the system constructs a local state summary:

$$z_{i,t} = (g_{i,t},\ r_{i,t},\ d_{i,t},\ \phi_{i,t},\ B_{i,t},\ h_{r,t}), \tag{2}$$

where $g_{i,t}$ is the current subtask goal, $r_{i,t}$ is the requested resource, $d_{i,t}$ is the expected occupation duration, $\phi_{i,t}$ captures task-conditioned execution features, $B_{i,t}$ is the remaining budget, and $h_{r,t}$ is the resource-side public state. In our implementation, $\phi_{i,t}$ includes the execution phase, prior retry count, and downstream dependency count; $h_{r,t}$ includes queue length and recent clearing prices. This summary provides a minimal decision interface. The full conversation history and global system state are not exposed to the allocator. Given $z_{i,t}$, the agent generates a structured action

$$x_{i,t} = (r_{i,t},\ d_{i,t},\ \beta_{i,t}) \tag{3}$$

via constrained prompting. The prompt enforces the resource type, budget bound, and JSON output schema (Appendix C). Before entering $\mathcal{Q}_t$, each action is deterministically validated for parseability, resource feasibility, duration feasibility, and budget feasibility. Only validated requests proceed to the clearing stage.

This stage implements the demand-side part of the mechanism. It turns local task information into a comparable bid, while the budget constraint makes excessive demand expression costly across future clearing rounds.

4.3 Stage 2: Optimal-Stopping Online Clearing

For each resource r, the runtime maintains a pending request set $\mathcal{Q}_t^r$ and an active occupation set $\mathcal{O}_t^r$. The currently allocable capacity is

$$\tilde{c}_r(t) = c_r - |\mathcal{O}_t^r|. \tag{4}$$

If a request arrives when idle capacity exists and no other request is pending for the same resource, access is granted directly without an auction. The clearing controller intervenes only when multiple eligible requests contend for the same resource.

Candidate window. Suppose contention for resource r is detected at time τ_0. The mechanism opens a candidate clearing window and accumulates validated requests into $\mathcal{C}_\tau^r$. At each inspection time τ, the controller compares clearing now with waiting one more interval Δ. Let $\mathrm{Top}_k(\mathcal{S})$ denote the set of the top-k requests in $\mathcal{S}$ ranked by bid.

The value of stopping is the total bid value that can be allocated immediately:

$$J_{\mathrm{stop}}(\tau) = \sum_{i \in \mathrm{Top}_{\tilde{c}_r(t)}(\mathcal{C}_\tau^r)} \beta_i. \tag{5}$$

The estimated value of waiting is

$$J_{\mathrm{wait}}(\tau, \Delta) = \widehat{\mathbb{E}}\left[\sum_{i \in \mathrm{Top}_{\tilde{c}_r(t)}(\mathcal{C}_\tau^r \cup \widehat{\mathcal{A}}_{\tau,\Delta}^r)} \beta_i \right] - \gamma_r \Delta \sum_{q_i \in \mathcal{C}_\tau^r} \beta_i. \tag{6}$$

Here $\widehat{\mathcal{A}}_{\tau,\Delta}^r$ denotes the predicted arrivals for resource r during the next interval, estimated from recent resource-specific arrival and bid history. The second term is the bid-weighted blocking cost of keeping current candidates waiting. The coefficient $\gamma_r > 0$ controls the cost scale for delaying resource r.

The expectation is estimated from public resource-specific runtime statistics, including recent arrival intervals and submitted bids; implementation details are given in Appendix E. The window closes when

$$J_{\mathrm{stop}}(\tau) \geq J_{\mathrm{wait}}(\tau, \Delta), \tag{7}$$

or when the window age reaches the maximum waiting horizon $w_{\max}$. A small minimum horizon $w_{\min}$ is used to avoid splitting nearly simultaneous requests into different rounds.

Clearing and settlement. After the window closes, the candidate set is frozen and requests that have become invalid are removed. The remaining eligible set is denoted by $\mathcal{P}_e^r$ for clearing round e. The mechanism selects winners by first-price sealed-bid clearing:

$$W_e^M \in \arg \max_{W \subseteq \mathcal{P}_e^r,\, |W| \le \tilde{c}_r(t)} \sum_{i \in W} \beta_i^e. \tag{8}$$

Ties are broken by arrival time. The mechanism uses first-price settlement: each winner pays $p_i^e = \beta_i^e$, and each non-winner pays zero. The budget update is

$$B_{i,t+1} = B_{i,t} - p_i^e - \rho_i^e, \tag{9}$$

where $\rho_i^e \ge 0$ is an optional penalty for anomalous behavior. Immediate payment links current access to future competitive capacity: higher payments reduce the remaining budget and constrain future bidding capacity. Without a strictly positive lower bound on winning payments, budget depletion alone does not guarantee a uniform bound on repeated contested wins.

Occupation and release. For each winning request, the system creates an occupation record

$$o_{i,t} = (a_i,\ r,\ t_{\text{start}},\ d_{i,t},\ t_{\text{deadline}}). \tag{10}$$

The record remains active until normal completion, early release, timeout, or abnormal termination. Upon release, one unit of capacity becomes allocable again. If pending requests remain for the same resource, the clearing controller is invoked again.

4.4 Analytical Characterization and Scope

We summarize the two analytical facts used to interpret the mechanism and the experiments. Detailed proofs are provided in Appendix D. The results do not rely on truthful bidding or an equilibrium model.

Informative bids. Fix a contested clearing round e for resource r. For each request i, let v_i^e be its latent one-step value and β_i^e its submitted bid. Assume that bids are conservative and informative: there exists $\alpha \in (0, 1]$ such that

$$\alpha v_i^e \le \beta_i^e \le v_i^e, \qquad \forall i \in \mathcal{P}_e^r. \tag{11}$$

Let W_e^M be the mechanism winner set and $W_e^\star$ the value-optimal feasible winner set under the same capacity constraint.

Proposition 1 (Approximate allocative efficiency)**.** *Under Eq.* (11)*,*

$$\sum_{i \in W_e^M} v_i^e \ge \alpha \sum_{i \in W_e^\star} v_i^e. \tag{12}$$

This connects Stage 1 to the bid–criticality analysis in §5.4: informative task-conditioned bids make bid-ranked clearing closer to the value-optimal allocation.

Contention and budget persistence. When $|\mathcal{P}_e^r| \le \tilde{c}_r$, all eligible requests can be served, so request ranking cannot change the allocation. Thus, the mechanism is expected to matter mainly under contention. Across contested rounds, a fixed budget bounds the number of wins when winning payments have a strictly positive lower bound.

Proposition 2 (Budget-bounded persistence)**.** *Let $B_{i,0}$ be the initial budget of agent a_i, and let $p_{\min} > 0$ be a lower bound on the clearing price across contested rounds won by a_i. Under Eq.* (9)*, the number of contested rounds won by a_i within one episode is at most*

$$\left\lfloor \frac{B_{i,0}}{p_{\min}} \right\rfloor. \tag{13}$$

This result states only that repeated contested wins consume future bidding capacity. It is not a fairness theorem and does not establish truthful revelation, dominant-strategy incentive compatibility, or classical *ex post* individual rationality.

In our cooperative setting, agents are explicitly prompted to bid for requested resources based on task urgency, remaining budgets, and future needs. This design elicited positive winning bids in all evaluated episodes. The empirical validity of these conditions is examined in Sections 5.4 and 5.5.

5 Experiments

5.1 Experimental Setup

Runtime. We construct a controlled event-driven runtime with $N = 6$ agents, two shared resource types, and exclusive resource occupation: GPU test environments with capacity $c_{\mathrm{GPU}} = 2$ and CPU execution slots with capacity $c_{\mathrm{CPU}} = 3$. All agents start with budget $B_0 = 100$. We evaluate three scenarios, S1–S3, using the same validated request stream and occupation/release logic across all methods. Results are means and standard deviations over five random seeds unless otherwise stated.

Datasets. We use software-engineering benchmarks to instantiate resource-dependent multi-agent executions. S1 uses Commit0-lite [30] to create homogeneous validation-resource contention. S2 and S3 use tasks adapted from PaperBench Code-Dev [23], a lightweight PaperBench variant focused on code-development requirements. S2 induces heterogeneous GPU and CPU contention, while S3 organizes the same task source as dependent task graphs, so critical-path delays can propagate downstream. These task sources vary contention structure while keeping the same access-right abstraction.

Metrics and baselines. Task score (TS) is normalized to $[0, 1]$: unit-test pass rate for S1 and rubric-based replication score for S2–S3. Makespan (MS), p95 acquisition latency, and Gini over per-agent completion counts measure completion time, tail waiting time, and access concentration, respectively. Higher TS and Spearman ρ between bids and post-hoc offline task criticality are better; lower MS, latency, and Gini are better. ΔMS is relative to the stated comparison target; positive values indicate worse makespan when defined relative to Market or the full mechanism. For contention sweeps, $\kappa = N/\sum_r c_r$.

We compare Market with FCFS [4], Fixed Priority [16], Round Robin [10], and SJF [4, 22]. All baselines use the same runtime, request stream, and occupation/release logic as Market, but do not observe bids, budgets, offline criticality labels, or latent one-step values. Appendix E gives implementation details.

5.2 Primary Performance under Contention

This experiment tests whether market-based arbitration improves end-to-end execution under shared GPU and CPU contention. We compare Market with four non-market schedulers across S1, S2, and S3, covering homogeneous contention, heterogeneous resource contention, and dependency-propagating contention. We report TS for task completion quality and MS for wall-clock completion time. Table 1 and Figure 3 report the comparison.

Table 1: Primary metrics across three scenarios. TS = task score (higher is better); MS = makespan in seconds (lower is better). Results are mean ± std over 5 seeds.

	S1: Homogeneous		S2: Heterogeneous		S3: Dependent	
Method	TS	MS	TS	MS	TS	MS
FCFS	0.40±.04	1547.3±82.6	0.48±.04	1726.8±93.1	0.33±.05	5681.2±312.7
Fixed Priority	0.42±.03	1543.8±71.4	0.39±.05	2469.4±142.8	0.43±.04	4221.5±218.3
Round Robin	0.41±.04	1659.2±96.3	0.46±.04	2239.6±116.7	0.40±.05	4374.8±244.1
SJF	0.44±.03	1578.4±68.7	0.37±.05	3742.8±246.9	0.46±.04	4148.6±205.4
Market (Ours)	**0.46±.03**	**1463.7±57.2**	**0.50±.03**	**1684.1±80.2**	**0.48±.04**	**3943.8±174.6**

Results. For makespan, the strongest primary baselines are Fixed Priority on S1, FCFS on S2, and SJF on S3. Relative to these baselines, Market reduces MS by 5.2% on S1, 2.5% on S2, and 4.9% on S3. Relative to the strongest primary TS baseline in each scenario, Market improves TS by about two percentage points. These TS gains are modest because the method changes runtime access order, not the base task-solving model. Together, these results support the first evaluation question: market-based arbitration improves completion time under contention while preserving or modestly improving task progress.

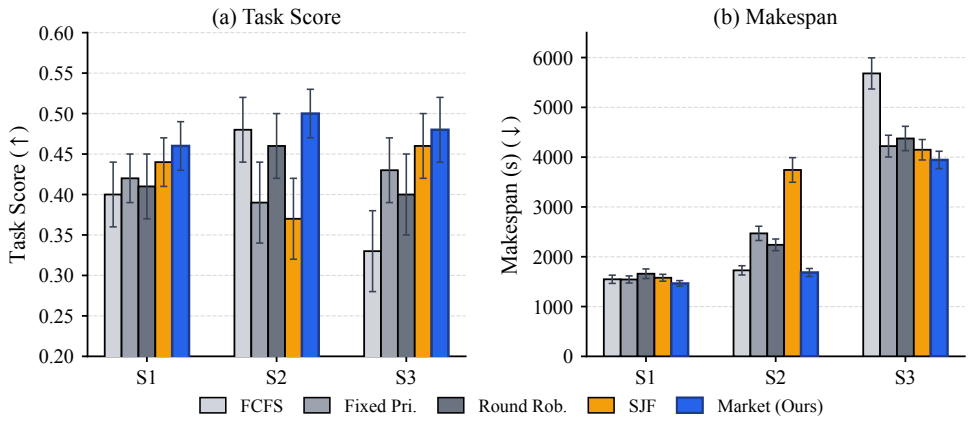


Figure 3: Primary performance across scenarios. Left: task score. Right: makespan. Market achieves the lowest makespan in all three scenarios while maintaining the best task score among the primary baselines.

5.3 Local-State Scorer Comparison

This experiment tests whether local-state features alone explain the gains. On S3, we compare Market with Rule Scorer and LLM Scorer, which use the same local-state features as Market to produce centralized priorities but do not use budget-constrained bids or intertemporal budget accounting. We report TS, MS, p95 acquisition latency, Gini, and ΔMS relative to Market.

Table 2: S3 comparison with centralized local-state scorers. TS = task score; MS = makespan in seconds; p95 = 95th percentile acquisition latency in seconds. Gini is computed over per-agent completion counts. ΔMS is relative to Market.

Method	TS	MS (s)	p95 (s)	Gini	ΔMS
FCFS	0.33±.05	5681.2±312.7	2.01±.15	0.50	+44.1%
SJF	0.46±.04	4148.6±205.4	2.00±.12	0.20	+5.2%
Rule Scorer	0.47±.04	4079.5±190.6	2.03±.11	0.27	+3.4%
LLM Scorer	0.48±.04	4008.9±181.7	2.01±.10	0.25	+1.7%
Market (Ours)	**0.48±.04**	**3943.8±174.6**	**2.00±.10**	**0.19**	0.0%

Results. Both scorers improve over non-market schedulers, but neither fully matches Market. Rule Scorer is +3.4% MS relative to Market, and LLM Scorer is +1.7% MS relative to Market. Market also has lower access concentration, with Gini $= 0.19$ versus 0.25 for LLM Scorer and 0.27 for Rule Scorer. This indicates that local-state features are useful, but the market format adds value by coupling current urgency expression to future budget capacity.

Matched value-aware controls. To assess the incremental role of the market protocol, we compare four S3 policies under the same request stream and OptimalStop clearing rule. Dynamic-CP+Stop uses decision-time critical-path priorities, LLM Scorer+Stop uses centralized local-state scores, CentralDebt+Stop adds intertemporal debt to those scores, and Market elicits agent-side bids with bid-dependent payments.

Market has 2.3% lower mean MS than CentralDebt+Stop, with a 49.0% reduction in allocator decision time and an 85.3% reduction in message volume. These contrasts jointly vary where task value is assessed and how cross-round costs are imposed; they do not isolate a single design feature. The LLM Scorer in Table 2 is the original baseline without OptimalStop and should be distinguished from LLM Scorer+Stop here.

Table 3: Matched S3 controls. Decision and message savings report Market's percentage reduction relative to the listed centralized method. MS values are descriptive means, without paired significance tests.

Method	MS (s)	Decision saving	Message saving
Dynamic-CP+Stop	4124.8	–	–
LLM Scorer+Stop	4058.7	66.7%	84.9%
CentralDebt+Stop	4036.9	49.0%	85.3%
Market (Ours)	**3943.8**	–	–

5.4 Bid Informativeness

This experiment tests whether demand elicitation produces bids that track request criticality. After execution, we compute offline criticality from realized critical-path membership and downstream blocking; this score is used only for post-hoc analysis and is not available to the online controller. We report Spearman correlation between bids and offline criticality and inspect budget trajectories by criticality group.

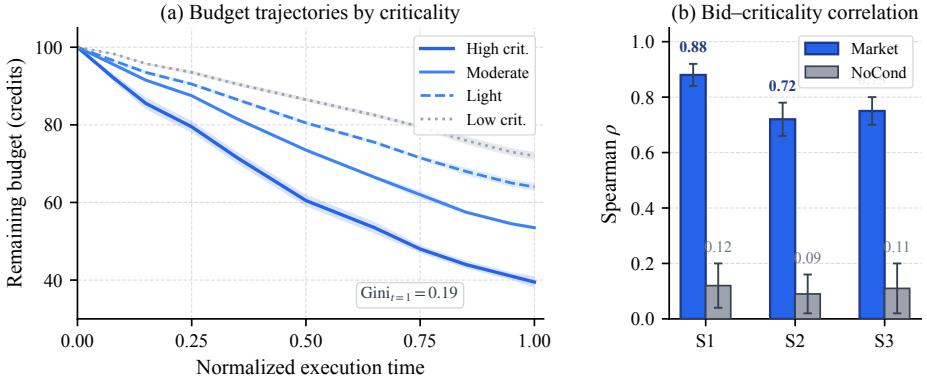


Figure 4: Mechanism behavior. Left: budget trajectories by task criticality on S3. Right: Spearman correlation between bids and offline task criticality, computed post-hoc from realized critical-path membership and downstream blocking. The patterns are consistent with task-conditioned bidding producing more informative demand signals than context-free bidding.

Results. Market bids have Spearman rank correlations of 0.88, 0.72, and 0.75 with offline task criticality on S1, S2, and S3, respectively. NoCond bids remain below 0.15 in all scenarios. The budget trajectories further show that high-criticality agents spend more aggressively, while lower-criticality agents conserve budget. These observations support the demand-elicitation claim: task-conditioned bidding converts local execution state into informative but budget-constrained demand signals.

Empirical validation of bid informativeness. To further examine the condition underlying Proposition 1, we estimate a proxy for latent request value using common-random-number replay. Across S1–S3, 99.4%–99.8% of requests satisfy the conservative upper bound in Eq. (11). The largest α achieving 90% request-level envelope coverage ranges from 0.81 to 0.86, while the complete envelope holds in 88.6%–92.8% of contested rounds. These results provide empirical evidence for the informativeness condition in the evaluated settings. The efficiency bound applies only to rounds satisfying its full premise. Detailed diagnostics are provided in Appendix F.8.

5.5 Component Ablation and Boundary Behavior

This experiment tests which mechanism components account for the gains and whether the gains increase with contention. The ablation study removes task-conditioned bidding, budget constraints, batching, the payment rule, or penalties while keeping the rest of the runtime fixed. The contention

sweep varies κ to change the number of agents relative to resource capacity. We report ΔMS for ablations and MS under different κ values.

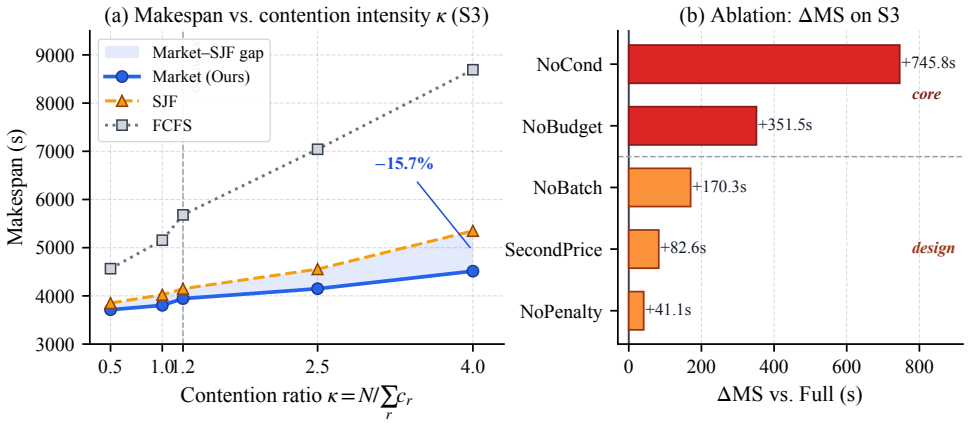


Figure 5: Component attribution and boundary behavior on S3. Left: makespan under increasing contention intensity. Right: makespan degradation from ablations. Core ablations account for larger degradation, while design-choice ablations are smaller.

Results. Removing task-conditioned bidding produces the largest degradation, increasing S3 MS by 745.8 seconds. Removing the budget constraint gives the second-largest degradation, increasing S3 MS by 351.5 seconds. The design-choice ablations are smaller: NoBatch increases MS by 170.3 seconds, SecondPrice by 82.6 seconds, and NoPenalty by 41.1 seconds. At $\kappa = 4.0$, Market reduces MS by 7.8% relative to LLM Scorer, 15.7% relative to SJF, and 48.1% relative to FCFS. These results indicate that task-conditioned bids and budget constraints are the main contributors, and that bid-ranked allocation matters most in contested rounds.

Empirical budget-bound diagnostics. We further examine the budget-bounded persistence condition in Proposition 2. The minimum observed winning payments are 0.78, 0.77, and 0.86 credits in S1–S3, respectively. With the initial budget $B_0 = 100$, these values yield sample-path bounds of 128, 129, and 116 contested wins. Since the current mechanism permits zero bids and imposes no positive reserve price, these observed bounds do not establish a uniform guarantee across execution paths. The budget mechanism's effect on access concentration is therefore supported by the reported empirical results within the evaluated settings.

6 Conclusion

We studied shared runtime access arbitration in LLM multi-agent systems, where agents contend for limited execution resources and request value depends partly on local task states. The proposed two-stage protocol elicits budget-feasible bids from agents and resolves asynchronous contention through optimal-stopping-based online clearing. Controlled experiments show lower makespan and modest task-score gains over non-market schedulers. Matched S3 controls further indicate reduced allocator decision latency and message volume relative to centralized value-aware scoring, with modest additional makespan gains. The observed benefits are strongest under moderate-to-high contention. Limitations include controlled cooperative workloads, a compact state interface, fixed non-replenishing budgets, and the lack of a mechanism-wide positive winning-price floor. Future work will examine budget recalibration, learned and strategic bidding, larger agent populations, additional LLM backbones, and production deployments.

Acknowledgments and Disclosure of Funding

Funding. This work was supported by the Key Research and Development Program of Wuhan City (Project No. 2025050602030055).

Competing Interests. The authors declare no competing interests.

References

[1] David Abramson, Rajkumar Buyya, and Jonathon Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18(8):1061–1074, 2002. doi: 10.1016/S0167-739X(02)00085-7.

[2] Alfonso Amayuelas, Jingbo Yang, Saaket Agashe, Ashwin Nagarajan, Antonis Antoniades, Xin Eric Wang, and William Wang. Self-resource allocation in multi-agent LLM systems. *arXiv preprint arXiv:2504.02051*, 2025. doi: 10.48550/arXiv.2504.02051.

[3] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *International Conference on Learning Representations (ICLR)*, 2024.

[4] Jr. Coffman, Edward G. and Leonard Kleinrock. Computer scheduling methods and their countermeasures. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, pages 11–21. ACM, 1968. doi: 10.1145/1468075.1468078.

[5] Michal Feldman, Kevin Lai, and Li Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *Proceedings of the 6th ACM Conference on Electronic Commerce (EC)*, pages 127–136, 2005. doi: 10.1145/1064009.1064023.

[6] Dawei Gao, Zitao Li, Xuchen Pan, Weirui Kuang, Zhijian Ma, Bingchen Qian, Fei Wei, Wenhao Zhang, Yuexiang Xie, Daoyuan Chen, Liuyi Yao, Hongyi Peng, Zeyu Zhang, Lin Zhu, Chen Cheng, Hongzhu Shi, Yaliang Li, Bolin Ding, and Jingren Zhou. AgentScope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*, 2024. doi: 10.48550/arXiv.2402.14034.

[7] Friedrich A. Hayek. The use of knowledge in society. *The American Economic Review*, 35(4): 519–530, 1945.

[8] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

[9] Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Ziyu Ye, Bowei Xia, Tao Sun, Zhaoxuan Jin, Yingru Li, Zeyu Zhang, Yifeng Wang, Qianshuo Ye, Bernard Ghanem, Ping Luo, and Guohao Li. OWL: Optimized workforce learning for general multi-agent assistance in real-world task automation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Leonard Kleinrock. Analysis of a time-shared processor. *Naval Research Logistics Quarterly*, 11(1):59–73, 1964. doi: 10.1002/nav.3800110105.

[11] Kevin Lai, Lars Rasmusson, Eytan Adar, Stephen Sorkin, Li Zhang, and Bernardo A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *arXiv preprint cs/0412038*, 2004. doi: 10.48550/arXiv.cs/0412038.

[12] Marco Laju, Donghyun Son, Saurabh Agarwal, Nitin Kedia, Myungjin Lee, Jayanth Srinivasa, and Aditya Akella. Nalar: An agent serving framework. *arXiv preprint arXiv:2601.05109*, 2026. doi: 10.48550/arXiv.2601.05109.

[13] Lior Levy, Liad Blumrosen, and Noam Nisan. On-line markets for distributed object services: The MAJIC system. In *3rd USENIX Symposium on Internet Technologies and Systems (USITS 01)*. USENIX Association, 2001.

[14] Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. In *International Conference on Learning Representations (ICLR)*, 2025.

[15] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 51991–52008, 2023.

[16] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. doi: 10.1145/321738.321743.

[17] Michael Luo, Xiaoxiang Shi, Colin Cai, Tianjun Zhang, Justin Wong, Yichuan Wang, Chi Wang, Yanping Huang, Zhifeng Chen, Joseph E. Gonzalez, and Ion Stoica. Autellix: An efficient serving engine for LLM agents as general programs. *arXiv preprint arXiv:2502.13965*, 2025. doi: 10.48550/arXiv.2502.13965.

[18] Kai Mei, Xi Zhu, Wujiang Xu, Wenyue Hua, Mingyu Jin, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. AIOS: LLM agent operating system. In *Conference on Language Modeling (COLM)*, 2025. doi: 10.48550/arXiv.2403.16971.

[19] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, 2024. doi: 10.18653/v1/2024.acl-long.810.

[20] Ying Sheng, Shiyi Cao, Dacheng Li, Banghua Zhu, Zhuohan Li, Danyang Zhuo, Joseph E. Gonzalez, and Ion Stoica. Fairness in serving large language models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 965–988, 2024.

[21] Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: A computational economy-based job scheduling system for clusters. *Software: Practice and Experience*, 34(6):573–590, 2004. doi: 10.1002/spe.581.

[22] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1–2):59–66, 1956. doi: 10.1002/nav.3800030106.

[23] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 56843–56873. PMLR, 2025.

[24] Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 173–191, 2024.

[25] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18(2):103–117, 1992. doi: 10.1109/32.121753.

[26] Yaoxiang Wang, Zhiyong Wu, Junfeng Yao, and Jinsong Su. TDAG: A multi-agent framework based on dynamic task decomposition and agent generation. *Neural Networks*, 185:107200, 2025. doi: 10.1016/j.neunet.2025.107200.

[27] Michael P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993. doi: 10.1613/JAIR.2.

[28] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling (COLM)*, 2024.

[29] Mingyan Yang, Guanjie Wang, Manqi Luo, Yifei Liu, Chen Chen, Han Zhao, Yu Feng, Quan Chen, and Minyi Guo. Justitia: Fair and efficient scheduling of task-parallel LLM agents with selective pampering. *arXiv preprint arXiv:2510.17015*, 2025. doi: 10.48550/arXiv.2510.17015.

[30] Wenting Zhao, Nan Jiang, Celine Lee, Justin T. Chiu, Claire Cardie, Matthias Gallé, and Alexander M. Rush. Commit0: Library generation from scratch. In *International Conference on Learning Representations (ICLR)*, 2025.

A Detailed Related Work

A.1 LLM multiagent systems and indirect access arbitration

Collaboration frameworks. LLM multiagent collaboration frameworks define how agents communicate, divide roles, and proceed through a workflow. AutoGen [28] supports programmable conversation patterns among agents. MetaGPT [8] expresses standard operating procedures through prompt sequences. AgentScope [6] provides a platform for building and managing multiagent applications, while ChatDev [19] organizes software development agents through communication protocols. AgentVerse [3] and CAMEL [15] study role based communication and collaborative behavior among LLM agents. These systems are important for organizing agent activity, but the resource access order usually follows workflow position, message order, or system level dispatch. They do not ask an agent to express the task value of a particular GPU test, CPU execution slot, or exclusive tool session. Thus, they can coordinate who should act next, but they do not directly solve request level access arbitration for scarce execution resources.

Task decomposition and planning. A second line of work assigns subtasks to agents or constructs execution plans. Amayuelas et al. [2] study strategies for distributing computational tasks among workers. Li et al. [14] decompose user queries and match subtasks to expert agents. OWL [9] uses a planner, coordinator, and worker hierarchy for task automation, and TDAG [26] generates specialized subagents for dynamically decomposed tasks. These methods reason about task structure and agent capability. Their allocation object is a subtask or an agent role, not an access right to a shared runtime resource. As a result, the importance of a later resource request is inherited from the plan or dependency graph only indirectly. When two agents request the same resource during execution, the planner does not necessarily know which request has larger marginal value for current task progress.

Runtime and serving systems. Agent runtime systems and LLM serving systems manage the physical resources that agents consume. Justitia [29] schedules task parallel LLM agents on shared GPU servers using memory cost and virtual time fairness. Autellix [17] treats agentic programs as serving objects and supports program aware preemption. Nalar [12] separates workflow specification from execution through a two level control architecture. AIOS [18] abstracts scheduling, context management, and access control into an operating system layer for LLM agents. Llumnix [24] studies dynamic scheduling for LLM serving, and Sheng et al. [20] formalize fairness in serving through virtual token counters. These systems act on the same physical substrate as our work, such as GPU capacity, CPU slots, memory, and request queues. They mainly rely on runtime records, including arrival time, expected duration, queue state, memory demand, capacity, or fairness counters. Such records are necessary for safe execution, but they are not sufficient to capture the task value of a request. The missing information often lies in the local state of the requesting agent: its current plan, execution phase, downstream blocking, retry pressure, and expected duration cost.

Position of this work. The above work provides the infrastructure within which shared execution resources are used. It also shows why central workflow or runtime records are not enough for the problem studied here. The central component can observe requests, queues, and resource occupation, but it may not know why a request matters for the agent's current task. This paper therefore isolates shared resource access arbitration as a separate problem. Instead of treating resource access as a byproduct of workflow order, task assignment, or runtime scheduling, it introduces a protocol through which agents submit constrained demand signals for their current requests.

A.2 Market mechanisms for distributed demand signals

Markets and dispersed knowledge. Hayek's analysis of dispersed knowledge argues that local conditions in complex systems cannot be fully collected and computed by a central mechanism in time [7]. A market can compress local judgments into comparable signals through bids and prices. This observation matches the information structure of LLM multiagent runtimes. The value of a shared resource request is not stored only in the orchestrator's task graph, the request queue, or the occupation record. It is partly held by the agent that knows its current plan, local progress, downstream dependencies, and future need for the same resource.

Compute resource markets. Computational resource markets have studied how bids, prices, and budgets can allocate scarce compute resources. Spawn [25] implements a computational economy on heterogeneous workstations. Wellman's market oriented programming [27] shows how market mechanisms can coordinate distributed allocation problems. Tycoon [11] implements market allocation for distributed clusters, and Feldman et al. [5] analyze efficiency and fairness in a fixed budget bidding game for shared clusters. Other compute economy systems, including Nimrod-G [1], Libra [21], and MAJIC [13], also use economic mechanisms to allocate distributed computational services. These works support the general idea that prices and bids can expose local demand to an allocator.

Why direct transfer is insufficient. LLM multiagent runtime arbitration differs from the settings above in two respects. First, the value of a resource request is not a fixed utility attached to a submitted job. It is generated during execution from the current plan, local task state, downstream dependency structure, retry pressure, and occupation duration. The mechanism therefore needs an elicitation interface that turns local task information into a comparable bid while enforcing a budget constraint. Second, resource requests arrive asynchronously and access has nonzero occupation duration. If the mechanism clears immediately after the first request, it may miss a later request that has higher task value. If it waits too long, it may block the current request and delay downstream work. Prior compute market systems do not jointly address agent generated demand, budget constrained expression, asynchronous bid arrival, and occupation tracking until release. This paper adapts the market view to this setting by combining structured bid elicitation with an online clearing rule for candidate windows.

B Broader Impact Statement

This work proposes a resource allocation mechanism for LLM multi-agent systems. The primary intended use is improving computational efficiency in shared-resource environments. We identify the following potential impacts:

Positive impacts. Market-based allocation can reduce task completion time under shared-resource contention, potentially improving computational efficiency. In the evaluated cooperative workloads, budget-constrained bidding is associated with reduced access concentration. General fairness guarantees and energy-efficiency effects require further evaluation.

Potential risks. If deployed in systems where agents represent different users or stakeholders, the mechanism introduces economic dynamics (bidding, budget depletion) that could disadvantage agents with less effective bidding strategies. In adversarial settings, agents could potentially manipulate bids to starve competitors. The mechanism currently assumes cooperative agents operating within a single organization; extension to multi-stakeholder settings would require additional safeguards against strategic manipulation.

Scope. This work studies controlled simulation environments. We do not deploy the mechanism in production or in contexts where allocation decisions directly affect end users.

C Demand Elicitation Prompt Template

The following is the structured prompt used by each agent to generate a bid during Stage 1. Bracketed fields are populated at runtime from the local state summary $z_{i,t}$ (Equation 2).

```
You are an autonomous task-executing agent in a multi-agent system.
```

```
You must request access to a shared runtime resource.

=== Your current task context ===
Subtask goal      : {g_i_t}
Execution phase   : {phi_phase}  (e.g., "initial", "retry_1", "final")
Downstream tasks  : {phi_downstream}  (count of tasks blocked by this step)
Prior retry count : {phi_retry}

=== Resource information ===
Resource type     : {r_i_t}  (e.g., "gpu_test_env", "cpu_exec_slot")
Estimated duration: {d_i_t} seconds
Queue length      : {h_queue}  (pending requests for this resource)
Recent avg. price : {h_price}  (clearing price from last 5 rounds)

=== Your budget ===
Remaining budget  : {B_i_t} credits
Maximum allowed bid: {B_i_t} credits

=== Instructions ===
Based on the above context, decide how urgently you need this resource
relative to your remaining budget and future needs.
Output ONLY valid JSON in the following schema (no other text):

{
  "resource": "<resource type string>",
  "duration": <estimated seconds, positive float>,
  "bid": <your bid in credits, float in [0.0, budget_max]>
}

Bid high if this step is on your critical path or blocking many
downstream tasks. Bid conservatively if this is an optional step
or you expect to need the resource again soon.
```

Requests that fail to parse or violate the budget constraint are rejected by the deterministic validator described in Section 4.2. In practice, the validation failure rate was below 1.2% across all experiments; rejected requests are requeued after a 500 ms backoff.

D Proofs for Analytical Characterization

This appendix provides the proofs for the two propositions stated in §4.4. All results are stated for a fixed resource type r and a fixed clearing round e. The eligible set is $\mathcal{P}_e^r$, the allocable capacity is $\tilde{c}_r$, the latent one-step value of request i is v_i^e, and the submitted bid is β_i^e.

D.1 Proof of Proposition 1

Proof. The mechanism selects W_e^M by maximizing total submitted bids over all feasible winner sets:

$$W_e^M \in \arg\max_{W \subseteq \mathcal{P}_e^r,\, |W| \le \tilde{c}_r} \sum_{i \in W} \beta_i^e.$$

Since $W_e^\star$ is feasible under the same capacity constraint, the bid optimality of W_e^M implies

$$\sum_{i \in W_e^M} \beta_i^e \ge \sum_{i \in W_e^\star} \beta_i^e. \tag{14}$$

By the upper bound in Eq. (11), $\beta_i^e \le v_i^e$ for every $i \in \mathcal{P}_e^r$. Therefore,

$$\sum_{i \in W_e^M} v_i^e \ge \sum_{i \in W_e^M} \beta_i^e. \tag{15}$$

By the lower bound in Eq. (11), $\beta_i^e \ge \alpha v_i^e$ for every $i \in \mathcal{P}_e^r$. Therefore,

$$\sum_{i \in W_e^\star} \beta_i^e \ge \alpha \sum_{i \in W_e^\star} v_i^e. \tag{16}$$

Combining Eqs. (15), (14), and (16) gives

$$\sum_{i \in W_e^M} v_i^e \geq \sum_{i \in W_e^M} \beta_i^e \geq \sum_{i \in W_e^\star} \beta_i^e \geq \alpha \sum_{i \in W_e^\star} v_i^e.$$

This proves Eq. (12). □

D.2 Proof of Proposition 2

Proof. Let $\mathcal{E}_i$ denote the set of contested clearing rounds in which agent a_i wins. By assumption, every such win has payment at least $p_{\min} > 0$. Thus,

$$\sum_{e \in \mathcal{E}_i} p_i^e \geq |\mathcal{E}_i| p_{\min}. \tag{17}$$

The budget update subtracts nonnegative payments and nonnegative penalties. Since budgets are not replenished and budget feasibility prevents negative budgets, the total amount charged to agent a_i over the episode cannot exceed its initial budget:

$$\sum_e (p_i^e + \rho_i^e) \leq B_{i,0}. \tag{18}$$

Because $\rho_i^e \geq 0$, Eq. (18) implies

$$\sum_{e \in \mathcal{E}_i} p_i^e \leq B_{i,0}. \tag{19}$$

Combining Eqs. (17) and (19) gives

$$|\mathcal{E}_i| p_{\min} \leq B_{i,0}.$$

Therefore,

$$|\mathcal{E}_i| \leq \frac{B_{i,0}}{p_{\min}}.$$

Since $|\mathcal{E}_i|$ is an integer,

$$|\mathcal{E}_i| \leq \left\lfloor \frac{B_{i,0}}{p_{\min}} \right\rfloor.$$

□

E Implementation Details

Agent framework and LLM backbone. All LLM-based agents use the same backbone model across all methods and scenarios, served through the OpenAI Responses API endpoint `https://api.openai.com/v1/responses` with model identifier `gpt-5.4`. For task-solving calls, we set temperature to 0.2, top-p to 1.0, maximum output length to 8192 tokens, and frequency and presence penalties to 0. For bid-generation calls, we use the same endpoint and model with temperature 0, top-p 1.0, maximum output length 300 tokens, and frequency and presence penalties set to 0.

Clearing-window and stopping parameters. The controller uses a minimum local batching horizon $w_{\min} = 200$ ms to merge requests arriving within one local round-trip interval. After this minimum horizon, the controller evaluates the stopping rule in Eq. (7) every Δ until the window closes or reaches the maximum waiting horizon $w_{\max}$. For each resource type, the controller maintains a rolling history of recent completed clearing windows. This history records request arrival times, submitted bids, window opening times, and window closing times. The controller uses these records to estimate the short-horizon arrival intensity and empirical bid distribution used in Eq. (6). The stopping rule uses only public runtime information: pending bids, recent arrivals, recent bids, allocable capacity, and current occupation state. It does not use latent one-step values or offline criticality labels. The NoBatch ablation sets $w_{\min} = 0$ and forces immediate clearing for each arriving request, removing local candidate-window accumulation.

Validation and occupation records. Before entering the pending set, every action is checked for parseability, resource feasibility, duration feasibility, and budget feasibility. For each winning request, the runtime creates an occupation record

$$o_{i,t} = (a_i,\ r,\ t_{\text{start}},\ d_{i,t},\ t_{\text{deadline}}).$$

The record is removed after normal completion, early release, timeout, or abnormal termination. Upon release, the corresponding unit of capacity becomes allocable again.

Duration estimation. The expected duration $d_{i,t}$ reported in each bid is computed by the agent as the mean of observed durations for the same subtask type in its own execution history, initialized to a task-type-specific prior. This estimate is used by the SJF baseline as well, ensuring a level comparison.

Baseline implementation. All non-market baselines use the same event-driven runtime and the same validated resource-request stream as Market. At each scheduling decision for resource r, a baseline observes the requested resource type, request arrival time, expected occupation duration $d_{i,t}$, current pending requests, active occupation records, and currently allocable capacity $\tilde{c}_r(t)$. The baselines do not observe submitted bids, remaining budgets, offline task criticality labels, or latent one-step values. They also do not use candidate clearing windows or predicted future arrivals. If the number of pending valid requests for a resource does not exceed the currently allocable capacity, all pending valid requests are granted access. Ranking rules are applied only in contested rounds. All baselines are non-preemptive: once a request receives access, it occupies the resource until normal completion, early release, timeout, or abnormal termination, using the same occupation-record logic as Market.

FCFS. FCFS represents arrival-order queue scheduling. For each resource, it maintains a first-in-first-out queue of valid pending requests ordered by arrival time. At a contested scheduling decision, FCFS grants access to the earliest requests up to the currently allocable capacity. Ties are broken by agent index. FCFS uses neither agent identity as a priority signal nor any duration, dependency, bid, budget, or criticality information. This baseline tests whether simple request-arrival order is sufficient for shared runtime access arbitration.

Fixed Priority. Fixed Priority represents static priority scheduling based on a fixed ordering of agents. At the beginning of each episode, the priority order is fixed as $a_1 \succ a_2 \succ \cdots \succ a_N$ and is kept unchanged across resources, scenarios, and random seeds. At a contested scheduling decision, pending requests are ranked by this fixed agent order. If the same agent has multiple pending valid requests for the same resource, the earliest request is considered first. Remaining ties are broken by arrival time and then by agent index. The fixed order is not updated using retries, downstream dependencies, queue outcomes, bids, budgets, or offline criticality. This baseline captures the class of access policies where priority is inherited from a static workflow or agent ordering rather than from the current request value.

Round Robin. Round Robin represents turn-taking fairness scheduling. For each resource, the runtime maintains a cyclic pointer over the fixed agent order $(a_1, \ldots, a_N)$, initialized to a_1 at the beginning of each episode. At a contested scheduling decision, the policy scans agents from the current pointer position, selects the earliest pending valid request from each visited agent, and skips agents with no pending valid request for the contested resource. The scan continues until the allocable capacity is filled or all agents have been visited. After the decision, the pointer advances to the next agent after the last selected winner; if no request is selected, the pointer is unchanged. Round Robin does not rank requests by expected duration, downstream blocking, submitted bid, remaining budget, or offline criticality. This baseline isolates turn-taking fairness from task-state-conditioned demand elicitation.

SJF. SJF represents duration-aware scheduling. At a contested scheduling decision, SJF ranks pending valid requests by the expected occupation duration $d_{i,t}$ submitted with the request and grants access to shorter requests first. Ties are broken by arrival time and then by agent index. The duration estimate is the value available at request time, computed as described in the duration-estimation paragraph above; SJF does not use realized completion time or any hindsight duration. SJF also does not observe submitted bids, remaining budgets, downstream dependency counts beyond their effect on the request, offline criticality labels, or latent one-step values. This baseline tests whether a visible resource metric, expected occupation duration, is sufficient without eliciting task-conditioned demand signals.

Budget initialization and accounting. All agents begin with $B_0 = 100$ credit units. Budget is not replenished during execution. Penalty charges $\rho_{i,t}$ are set to 10 credits for no-show and 5 credits for overrun exceeding 50% of the declared duration. Penalties are capped by the remaining budget after payment, so budget feasibility is preserved. In the NoPenalty ablation, $\rho_{i,t} = 0$ throughout.

Reproducibility. The primary five-seed experiments use fixed random seeds (0–4) for task graph generation and stochastic occupation durations. Each full experimental run, including all methods, all scenarios, and 5 seeds, completes in approximately 6 hours on a workstation with 2 NVIDIA A100 GPUs and 64 CPU cores. Code is available at `https://github.com/a7c70200g/MARA`.

F Supplementary Results

This appendix provides supplementary experimental results for Section 5. We report paired significance tests, secondary metrics, exact ablation values, online clearing-window ablations, critical-path waiting decomposition, budget sweeps, contention sweeps, and overhead measurements.

F.1 Paired Significance Tests

Table 4 reports paired tests over 30 seeds. These paired tests are separate from the five-seed main tables, so means may differ slightly from Table 1. The gains are modest in absolute magnitude but directionally stable, especially under dependent and high-contention workloads.

Table 4: Paired significance tests over 30 seeds.

Scenario	Baseline	Market MS	Base MS	ΔMS	p-value
S1	Fixed Priority	1465.2 ± 61.3	1546.1 ± 76.0	-80.9	.004
S2	FCFS	1685.8 ± 82.4	1724.0 ± 91.5	-38.2	.031
S3	SJF	3947.6 ± 179.8	4145.9 ± 211.2	-198.3	.002
S3, $\kappa = 4.0$	LLM Scorer	4513.4 ± 244.1	4894.8 ± 278.7	-381.4	$< .001$

Table 5 reports task-score tests and multiple comparisons. Using the same 30 paired seeds, the three primary TS effects are evaluated with two-sided paired permutation tests. The 95% bootstrap confidence intervals for all three TS effects exclude zero. Applying the Holm procedure jointly to the three main MS and three TS tests gives adjusted MS p-values of 0.008, 0.031, and 0.008 on S1–S3, respectively.

Table 5: Paired task-score tests over 30 seeds. ΔTS is the paired Market–baseline difference; Holm-6 jointly adjusts the three primary MS and three TS comparisons.

Scenario	Baseline	ΔTS	Raw p	Holm-6 p
S1	SJF	$+0.025$	.001	.006
S2	FCFS	$+0.042$	.002	.008
S3	SJF	$+0.048$	.001	.006

F.2 Secondary Metrics

Table 6 reports secondary metrics on S3. Market has the lowest median acquisition latency, matches the lowest rounded p95, and has the lowest contention count and clearing overhead. Lower utilization in this DAG setting reflects reduced non-critical occupation rather than poor use of a continuously available workload, since predecessor dependencies force idle intervals.

Table 6: Secondary metrics on S3. Util = fraction of capacity-time occupied; p50 and p95 = median and 95th percentile acquisition latency in seconds; Gini = Gini coefficient over per-agent completion counts; Contention = total contention events; OH = clearing overhead in seconds.

Method	Util	p50 (s)	p95 (s)	Gini	Contention	OH (s)
FCFS	.059±.01	0.19±.02	2.01±.15	0.50	12	0.12
Fixed Priority	.020±.005	0.12±.01	2.64±.20	0.23	9	0.04
Round Robin	.029±.006	0.15±.02	2.01±.15	0.18	12	0.06
SJF	.021±.005	0.13±.01	2.00±.12	0.20	10	0.04
Market (Ours)	.017±.004	**0.10±.01**	**2.00±.10**	0.19	**4**	**0.03**

F.3 Exact Ablation Values

Table 7 reports the exact ablation values summarized in Figure 5. The two core ablations dominate degradation, while design-choice ablations produce smaller changes.

Table 7: Ablation results on S3. ΔTS = change in task score relative to the full mechanism; ΔMS = change in makespan relative to the full mechanism.

Variant	TS	MS (s)	ΔTS	ΔMS
Full mechanism	0.48	3943.8	–	–
NoCond	0.44	4689.6	−0.04	+745.8
NoBudget	0.46	4295.3	−0.02	+351.5
SecondPrice	0.47	4026.4	−0.01	+82.6
NoBatch	0.47	4114.1	−0.01	+170.3
NoPenalty	0.48	3984.9	0.00	+41.1

F.4 Online Clearing Policy Ablation

The NoBatch row in Table 7 shows that immediate clearing is worse than the full mechanism, but it does not by itself distinguish optimal stopping from ordinary batching with a fixed window. Table 8 therefore holds the bidding rule, payment rule, and budget accounting fixed, and varies only the clearing-window policy. Value denotes the normalized allocated bid value per contested clearing round. Miss is the fraction of rounds in which a request arriving shortly after window closure would have entered the allocable top set.

ImmediateClear misses many late high-value requests and increases makespan by 4.3%. Fixed windows improve over immediate clearing, but their performance depends on the chosen horizon: short windows still miss valuable arrivals, while long windows increase critical-path waiting. NoStop obtains high allocated value by waiting until $w_{\max}$, but the extra blocking dominates and raises makespan by 6.9%. OptimalStop achieves the lowest feasible makespan and critical-path waiting, while remaining within 1.3% of OracleWindow, which selects the best window with hindsight. This supports the claim that the online clearing gain comes from adaptive stopping rather than batching alone.

Table 8: Ablation of online clearing-window policies on S3. Age = clearing-window age; CP wait = critical-path waiting time; Value = normalized allocated bid value per contested clearing round; Miss = fraction of rounds in which a request arriving shortly after window closure would have entered the allocable top set; ΔMS = makespan change relative to OptimalStop. Fixed-window baselines isolate ordinary batching from the proposed optimal-stopping controller. OracleWindow uses hindsight future arrivals and is not an online method.

Policy	TS	MS (s)	Age (ms)	CP wait	Value	Miss	ΔMS
ImmediateClear	0.47±.04	4114.1±190.8	0±0	492±46	0.71±.06	24%	+4.3%
Fixed-200ms	0.47±.04	4048.7±184.9	200±0	438±41	0.78±.05	17%	+2.7%
Fixed-500ms	0.48±.04	3986.4±179.2	500±0	383±37	0.83±.05	10%	+1.1%
Fixed-1000ms	0.47±.04	4071.9±191.5	1000±0	412±39	0.84±.05	7%	+3.2%
NoStop	0.46±.04	4216.5±214.3	1200±0	469±48	0.86±.04	5%	+6.9%
OptimalStop	**0.48±.04**	**3943.8±174.6**	342±88	**361±35**	0.85±.04	8%	0.0%
OracleWindow	0.49±.04	3892.6±168.8	318±74	334±32	**0.88±.04**	**4%**	−1.3%

F.5 Critical-Path Protection

Table 9 decomposes waiting time on S3. Market reduces critical-path waiting and downstream blocked time relative to FCFS and SJF. NoBudget also prioritizes high-criticality requests, but does so with more concentrated access.

Table 9: Critical-path protection on S3. CP wait = waiting time for critical-path requests; Non-CP wait = waiting time for non-critical-path requests; Blocked = downstream blocked time.

Method	CP wait (s)	Non-CP wait (s)	Blocked (s)	Top-Q win
FCFS	742±68	221±31	1916±210	0.29±.05
SJF	516±49	164±22	1212±138	0.37±.06
NoCond	621±57	234±28	1540±173	0.31±.05
NoBudget	449±42	310±37	1004±112	0.78±.08
LLM Scorer	407±38	248±30	913±102	0.62±.07
Market (Ours)	**361±35**	281±32	**783±91**	0.66±.06

F.6 Budget Sweep

Table 10 reports the S3 budget sweep. Very small budgets limit urgency expression, while very large budgets weaken the scarcity signal. The default budget gives the best makespan and a low exhaustion rate.

Table 10: Budget sweep on S3. Moderate budgets provide a favorable tradeoff between urgency expression, makespan, and observed access concentration in the evaluated workloads.

Variant	TS	MS (s)	Win Gini	Max win	Exhaust.
$B_0 = 25$	0.45±.04	4312.6±208.4	0.22	0.31	28%
$B_0 = 50$	0.47±.04	4017.3±186.2	0.25	0.29	8%
$B_0 = 100$	0.48±.04	**3943.8±174.6**	0.28	**0.26**	3%
$B_0 = 200$	0.48±.04	4009.6±187.5	0.36	0.34	0%
NoBudget	0.46±.04	4295.3±201.8	0.55	0.48	N/A

F.7 Full Contention Sweep

Table 11 reports the full S3 contention sweep. At low contention, all work-conserving methods converge. As contention increases, Market increasingly outperforms non-market baselines and the LLM local-state scorer.

Table 11: Extended contention sweep on S3. Gain is reported relative to the LLM scorer.

κ	Market MS	LLM MS	SJF MS	FCFS MS	Gain
0.5	3602 ± 151	3596 ± 148	3665 ± 160	3710 ± 172	-0.2%
1.0	3818 ± 166	3865 ± 170	3982 ± 181	4930 ± 260	$+1.2\%$
1.2	3944 ± 175	4009 ± 182	4149 ± 205	5681 ± 313	$+1.6\%$
2.0	4172 ± 196	4328 ± 214	4620 ± 238	6550 ± 365	$+3.6\%$
3.0	4388 ± 221	4675 ± 251	5016 ± 284	7660 ± 431	$+6.1\%$
4.0	4513 ± 244	4895 ± 279	5349 ± 326	8692 ± 508	$+7.8\%$

F.8 Empirical Coverage of the Bid-Informativeness Condition

Common-random-number replay holds exogenous request streams and random draws fixed, while execution paths may differ. We calibrate a bid-independent value scale on separate seeds and freeze it for held-out evaluation. Table 12 reports the condition in Eq. (11); replay-value efficiency is 0.985–0.995. The guarantee applies only to complete rounds satisfying the envelope.

Table 12: Bid-envelope checks. Upper denotes $\beta_i^e \le v_i^e$; full round denotes satisfaction of the entire envelope at the reported α.

Scenario	Spearman ρ	Upper	α at 90%	Full round
S1	0.88	99.8%	0.86	92.8%
S2	0.72	99.4%	0.81	88.6%
S3	0.75	99.6%	0.85	90.7%

F.9 Bidder and Budget Diagnostics

Context coverage. On S3, full-context bids raise Spearman ρ from 0.75 to 0.86 and reduce execution-only MS by 1.8%, but raise end-to-end MS by 6.7% because of bid-generation cost.

Value-weighted budgets. In exploratory five-seed comparisons, changes in TS/MS are $+2.2\%/+1.5\%$ (S1), $-2.0\%/-1.7\%$ (S2), and $0\%/+1.2\%$ (S3). Positive MS improvement means lower makespan. Cross-seed standard deviations rise by about 50%–67%, with scenario-dependent effects.

F.10 Sensitivity to Bid Errors

On S3, bounded bid noise at levels 0/0.02/0.05/0.10 yields stable-round rates 100.0/99.2/97.0/92.1% and winner-change rates 0.0/0.3/1.2/3.6%; at 0.10, TS is unchanged and MS increases 0.56% (five paired seeds).

Aggressive bidding by 1/2/4 agents (up to +60%) raises MS by at most 1.2/2.3/3.6%, respectively. With four agents at +60%, selected-agent win share is 75.6% and budget exhaustion is 43%. These are empirical stress tests without an incentive-compatibility guarantee.

F.11 Overhead

Bid generation consumes $3.2\% \pm 0.6\%$ of wall-clock task time, and auction clearing consumes $0.38\% \pm 0.12\%$. The total measured overhead is $3.58\% \pm 0.71\%$. Because bid generation and clearing often overlap with waiting periods, the effective critical-path overhead is lower. All reported makespans include the measured bid-generation and clearing overhead.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the proposed market-based protocol, its scope in shared GPU/CPU runtime contention, and the main empirical claims. These claims are supported by the theoretical characterization in Section 4.4 and the controlled experiments in Section 5 and Appendix F.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the controlled simulation scope in the Broader Impact Statement and notes future work on larger agent populations, additional LLM backbones, strategic bidding behavior, and production deployment in the conclusion. A separate Limitations section could make these limitations more visible.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Section 4.4 states the assumptions for the two propositions, including the bid informativeness condition and the budget update condition. Complete proofs for both propositions are provided in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper discloses the information needed to reproduce the main experimental results that support the claims. Section 5.1 specifies the scenarios, resource capacities, baselines, metrics, budget setting, and random seeds, while Appendices C, E, and F provide the bid-generation prompt, clearing-window parameters, validation rules, duration estimation, budget accounting, compute setup, significance tests, ablations, budget sweeps, contention sweeps, and overhead measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The implementation and experiment code are publicly available at `https://github.com/a7c70200g/MARA`. It includes instructions for preparing the benchmark tasks, running all proposed-method and baseline experiments, reproducing the tables and figures in Section 5 and Appendix F, and verifying the reported logs under the fixed random seeds and compute setup described in Appendix E.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup, evaluation metrics, baselines and main implementation details are described in Section 5.1 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main results report mean and standard deviation over five random seeds, and Appendix F reports paired significance tests over 30 seeds with p-values. The figures and tables indicate variability for the main empirical comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the compute setup and approximate runtime needed to reproduce the experiments in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is evaluated in controlled simulation environments, does not involve human subjects, and does not deploy the mechanism in settings that directly affect end users. The paper also discusses potential societal risks and scope limitations in the Broader Impact Statement.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix B discusses positive impacts such as improved resource utilization and potential energy efficiency. It also discusses risks related to bidding dynamics, budget depletion, strategic manipulation, and multi-stakeholder deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a high-risk pretrained model, image generator, or scraped dataset. The proposed contribution is a runtime resource allocation mechanism evaluated in controlled simulation environments.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper credits the datasets, benchmarks, LLM backbone, and software assets used in the experiments, and the supplemental material documents their licenses, versions, and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new experimental assets released with the paper are documented in the supplemental material, including configuration files, reproduction scripts, and instructions for generating the reported results.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. All evaluations are conducted in controlled multi-agent runtime environments.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research. Therefore, IRB approval or equivalent human-subject review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper describes the use of LLM agents for bid generation in the proposed mechanism, including the local-state input, constrained JSON output format, validation procedure, and prompt template in Section 4.2 and Appendix C.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.