协同边缘计算中的可靠性感知的AI智能体执行：一种世界模型方法

Xiangchun Chen、Jiannong Cao，*IEEE*会士，Longxiang Gao、Xiaoming Wu 和 Yusheng Ji，*IEEE*会士

摘要—协同边缘计算通过将执行分配到资源受限的异构用户控制设备上，为AI智能体应用提供支持。本文研究节点故障和无线中断情况下的可靠性感知的AI智能体执行。这些中断可能导致智能体执行状态不可用或不一致，从而迫使系统进行恢复或重新计算，消耗剩余的截止期限预算。因此，维护可恢复执行状态有助于在截止时间前完成请求。我们提出一个在线AI智能体执行管理问题，在资源、兼容性和截止期限约束下，联合选择资源分配、状态保护操作和恢复目标，以最大化预期任务成功率。我们构建了一个世界模型，在候选管理操作下联合刻画不断演化的智能体工作流和可恢复执行状态，同时考虑节点可靠性、网络条件和资源可用性。该模型使控制器能够在候选动作下仿真未来的执行和恢复轨迹，并在截止时间前估计其对执行进度和任务完成的影响。随后，我们采用基于世界模型的模型预测控制（WorldMPC）结合蒙特卡洛展开，在动态环境下联合优化资源分配、状态保护和恢复决策，以实现长期可靠的智能体执行。我们使用真实数据集和合成数据集，在仿真环境和真实世界测试平台上对所提出的方法进行评估。实验结果表明，在仿真环境和真实世界测试平台中，WorldMPC均较基线方法提高了任务成功率。

索引词 **s**—协同边缘计算；AI智能体；可靠性；状态恢复；世界模型；模型预测控制 I

arXiv:2611.32520v1 [cs.DC] 24 Nov 2026

1 引言

AI智能体将语言模型推理与工具使用相结合，以在用户控制的基础设施上完成任务。协同边缘计算（CEC）使异构设备能够共享计算资源、内存、模型和工具，突破单个设备的能力限制 [1], [2]。节点故障和无线中断使得在截止时间前实现可靠完成成为核心问题。

对于有状态智能体执行而言，可靠性尤其具有挑战性。推理输出可能决定后续的工具操作，而工具执行结果可能揭示新的子任务、改变依赖关系，并增加剩余工作负载。恢复执行需要推理上下文、已完成子任务和工具结果、生成文件以及环境信息。故障可能摧毁这一状态，而断开连接可能使幸存状态暂时无法访问。恢复该状态或重构丢失的工作会消耗剩余的截止期限预算。保护最新状态可以减少恢复工作，但会与执行竞争内存与带宽。

现有研究分别探讨了智能体编排 [3], [4],推理状态保存与恢复 [5]–[7],以及截止时间与可靠性感知的工作流调度 [8]。剩余的挑战在于：当中间结果改变工作流、失败改变其可恢复进度时，如何协调这些能力。

• *Xiangchun Chen* 和 *Jiannong Cao* 就职于中国香港特别行政区香港理工大学计算系（电子邮箱：*xiang-chun.chen@connect.polyu.hk*；*jiannong.cao@polyu.edu.hk*）。

• *Longxiang Gao* 和 *Xiaoming Wu* 就职于中国济南的齐鲁工业大学（山东省科学院）山东省计算机科学中心（电子邮箱：*gaolx@sdas.org*；*wuxm@sdas.org*）。

• *Yusheng Ji* 就职于日本东京的国立信息学研究所（电子邮箱：*kei@nii.ac.jp*）。

我们研究协作式边缘计算中的可靠性感知的AI智能体执行。每个请求都遵循一个动态揭示的依赖性推理和工具子任务有向无环图（DAG）。可靠性控制器在计算资源、内存、带宽和兼容性约束下，联合确定资源分配、状态保护和恢复目标。我们的目标是最大化预期任务成功率：即在截止时间之前将完整且协议一致的输出发送到客户端的已提交请求的预期比例。

这个问题存在三项挑战。首先，动态依赖关系以及不确定的推理或工具执行结果，使剩余工作负载和执行时间难以预测。其次，有效的恢复需要推理进度、工具结果和文件之间保持一致状态。其
控制器必须决定保护哪些进度以及在哪里进行保护，同时考虑状态可用性和恢复执行所需的工作量。最后，资源分配、保护和恢复会随着时间产生耦合效果。这些效果必须根据不断变化的风险以及多个请求的剩余截止时间进行评估。

为应对这些挑战，我们提出一种基于世界模型的模型预测控制（WorldMPC）方法。我们的世界模型联合表示动态工作流进度和可用于一致恢复的执行状态。它预测资源分配、状态保护和恢复决策如何影响后续执行、状态可用性和恢复成本。学习得到的事件预测与显式执行规则共同生成遵循已提交执行历史的轨迹。在基于学习动力学的规划基础上， [9], WorldMPC 使用蒙特卡洛展开，根据估计的按时任务成功率比较可行的资源、保护和恢复联合计划。它执行所选计划的首个动作，并在

接收到新的观察后重新规划。

本文的主要贡献如下：

- **联合可靠性感知执行建模**。我们将智能体执行表述为一个联合资源分配、状态保护与恢复问题。该模型将动态揭示的依赖关系和可恢复进度，与共享资源及兼容性约束下的任务截止期限相联系。

- **面向可靠智能体执行的基于世界模型的规划**。我们构建了一个世界模型，用于联合表示动态工作流进度和可恢复的执行状态。该模型生成与已提交历史一致的执行轨迹和恢复轨迹。WorldMPC利用这些轨迹，根据估计的按时任务成功率比较联合资源、保护与恢复计划。

- **仿真与真实世界测试平台评估**。我们在仿真环境和真实世界测试平台上，使用真实世界数据集与合成数据集，将WorldMPC与最先进的基线方法进行比较。实验结果表明，我们的方法提升了任务成功率，优于基线方法。

本文其余部分的组织结构如下。第2节回顾相关工作。第3节介绍系统模型和问题表述。第4节阐述所提出的方法。第5节介绍实验评估，第6节对全文进行总结。

2 相关工作

我们从三个角度回顾相关工作：AI智能体执行与编排、容错执行与状态恢复，以及面向可靠性的调度。

2.1 AI智能体执行与编排

Yao 等人 [10] 提出了 ReAct，该方法交错进行推理、动作和观察，并利用环境反馈修正决策。然而，如何选择和使用外部工具仍是一项核心挑战。Schick等人 [11] 提出了 Toolformer，该方法能够学习何时调用 API、提供哪些参数，以及如何纳入 API 的输出。然而，学习调用工具本身并未提供保留任务尝试失败经验的机制。Shinn 等人 [12] 引入了 Reflexion，通过语言反馈和反思情景记忆来解决这一问题。这些方法改进了智能体轨迹中的工具选择和任务求解行为，而规模更大的协作工作流则需要额外的机制。

Wu等人 [13] 开发了 AutoGen，通过可编程对话将可定制的智能体与工具相结合，但开发人员必须指定合适的交互模式。Hong等人 [14] 提出了 MetaGPT，通过标准化操作流程、角色分配和中间产物来构建协作流程。这减少了软件开发工作流中的不一致性，但无法自动确定独立工具调用的高效顺序。Kim等人 [3] 使用 LLMCompiler 解决了这一问题，将函数调用规划、任务获取和并行执行分离开来。并行调度减少了串行化，而推理和动作的选择轨迹仍然是一个独立的规划问题。Zhou等人 [15]开发了语言智能体树搜索，将蒙特卡洛树搜索、自我反思和环境反馈相结合，以探索不同方案。其搜索关注的是任务求解轨迹，而不是设备放置，也不涉及基础设施故障后执行状态的存续。

Hu 等人 [16]提出了智能体系统自动化设计，其中元智能体对候选智能体进行编程，并利用以往发现的存档来指导进一步搜索。Zhang 等人 [4]开发了 AFlow，该方法使用蒙特卡洛树搜索和执行反馈，对代码表示的工作流进行搜索。这些方法减少了对人工设计智能体结构的依赖，但并未消除多智能体执行中的通信开销。为解决这一开销，Zhang 等人 [17]提出了 AgentPrune，该方法对时空消息传递图中的冗余通信进行剪枝。然而，它没有确定应如何在不可靠的本地设备之间调度剩余的计算和状态传输。我们关注的是这些流程所产生的执行进度：已提交的推理输出、工具结果，以及后续子任务所用的制品。

2.2 容错执行与状态恢复

Moritz 等人 [18] 提出了 Ray，该系统支持动态任务和执行器执行，并根据血缘关系重构丢失的任务输出。然而，当任务修改持久化应用状态时，它无法保证状态一致性。
Zhang 等人 [19] 针对有状态无服务器工作流，将日志记录、容错与事务执行相结合。Jia 等人 [20]开发了 Boki，通过公开共享日志来支持一致且持久的状态以及容错工作流。然而，这些运行时的恢复保证不会自动扩展到外部工具操作。

对于大语言模型推理，PagedAttention 可减少 KV 缓存碎片化并支持缓存共享 [21]，，但当工作节点不可用时，其自身并不会保留存活副本。Llumnix 会在模型实例之间迁移运行中的请求和内存中的状态，以实现负载均衡和隔离 [22]。ServerlessLLM 将检查点局部性、快速模型加载和实时推理迁移相结合，以降低启动延迟 [23]。然而，意外的源端丢失需要明确的恢复机制。

D´ej`aVu 使用 KV 缓存流式传输和复制，在发生失败后保留进度 [5]。SpotServe 根据可抢占 GPU 的可用性调整模型并行化，并提交推理进度以支持有状态恢复 [24]。然而，传输或复制大型推理状态会消耗稀缺的带宽。EdgeFlow 将直接 KV 传输与基于激活值的重新计算相结合，以缩短协同边缘推理中的迁移时间 [6]。然而，在选择恢复目的地时，它没有综合考虑迁移时间和恢复执行的请求所产生的执行负载。LUMEN 在模型重新加载期间，综合考虑检查点放置、中断请求分配和容量恢复 [7]。

对于智能体，已提交的推理输出可能决定后续的工具操作和依赖子任务。因此，恢复时必须使已保存的推理进度与已提交的工具执行结果、文件和环境变化保持一致。如果

重复调用模型产生了不同的输出，后续执行仍必须遵循已提交历史。

2.3 可靠性感知调度

Qin 等人提出了一种面向多云工作流的完工时间、成本和可靠性的多目标模因算法 [25]。然而，该方法在执行前确定备份放置，无法在发生失败后进行自适应任务恢复。RECR 面向受能源和截止时间约束的物联网工作流；RECR-D 通过增加自适应任务复制来容忍瞬态和永久故障 [26]。DARCS 针对异构边缘基础设施上的大语言模型智能体工作流，制定了受截止时间和可靠性约束的准入与调度方案；DARCS-H 则结合了可靠性调整优先级、可靠性预算跟踪和约束修复 [8]。然而，这些研究并未联合控制不断演化的智能体执行状态的保护与恢复，而新的工作会持续由中间结果产生。

AlpaServe 利用模型放置和并行度，在突发需求下发挥统计复用的优势 [27]。然而，它未考虑预填充和解码各自不同的资源需求。DistServe 将预填充与解码分离，并优化二者的资源分配和并行度 [28]，而 Sarathi-Serve 采用分块预填充和无停顿调度，以平衡吞吐量与延迟 [29]。然而，这些系统关注的是请求级服务，而不是应用工作流之间的依赖关系。Parrot 通过语义变量公开应用数据流，从而优化相互依赖的大语言模型请求 [30]，解决了这一局限。然而，它在调度决策中并未联合考虑执行状态保护和故障恢复。

TetriSched 将运行时估计和截止时间与集群条件变化时的反复重新规划相结合 [31]。然而，它并未对应用组件之间的性能依赖关系进行建模。Sinan 学习微服务依赖关系，以便在维持端到端尾时延目标的同时分配资源 [32]。对于多阶段无服务器工作流，AQUATOPE 使用贝叶斯模型进行容器预热和资源分配，以在不确定性下保障工作流级服务质量 [33]。然而，这些系统并未联合考虑预测性资源分配、执行状态保护和故障恢复。

随着智能体工作流不断演进，失败会改变可恢复的进度；保护已提交切面可以减少重复推理和工具操作，但复制和恢复该切面会消耗与正在进行的执行共享的资源。这种相互影响促使我们联合进行资源分配、状态保护和恢复，以便在节点和链路中断时最大化预期任务成功率。

3 系统模型与问题形式化

本节将具备可靠性感知的 AI智能体执行形式化为一个联合在线问题：推进演进中的工作流并 պահպան其可恢复状态。推理和工具执行结果既会揭示新的工作，也会创建状态依赖关系。我们对执行、保护和恢复如何共享资源，以及它们如何共同决定在截止时间前完成任务进行建模。表 1 总结了主要符号。

表 1 文中使用的主要符号

符号	描述
$\mathcal{N}, \mathcal{E}$	计算节点和通信链路
a_i, d_i, D_i	到达、目标时延和绝对截止时间
Ω_i	请求 i所需模型和工具集
$\mathcal{A}_i(t)$	当前已揭示的智能体工作流有向无环图
$\mathcal{V}_i(t)\ \ \mathcal{E}_i(t)$	已揭示子任务和依赖边
$W_i(t)$	请求 i的剩余工作负载
$\mathcal{C}_{iv}(t)$	请求 i的子任务 v 的兼容目标
$p_{iv}(t)\ x_{ivn}(t)$	权威执行器和二进制活动分配- ment
$n_i(t)$	已提交执行事件数量
$\mathcal{O}_i(k)\ V_i(k)$	切点 k 处的状态对象清单和逻辑字节数
$k_{in}^{\mathrm{bk}}(t)$	最大的完整兼容已确认切点保留在节点 n
$\delta_{in}(t)$	保留检查点的已提交事件滞后
$\mathcal{B}_{in}(t)\ \Delta V_{in}$	已验证的目标对象和增量复制字节
$\mathcal{K}_{in}^{\mathrm{avail}}(t)$	可供恢复访问的完整兼容切面目标 n
$\kappa_i(t)$	选定的恢复切面；零表示初始重构- 造，受重放可行性约束
$G_n^r(t)\ \gamma_{ivn}^r(t)$	池中的计算资源容量与子任务分配 r
$\gamma_{\cdot\ n}^{r\ \ \mathrm{mgmt}}(t)$	保护与恢复需求的计算资源
$M_{nh},\ W_{nh}$	存储容量与驻留模型权重占用- 量
$\mathcal{Q}_{nh}(t)$	层级 h中的不同已分配物理状态对象
$M_{nh}^{\mathrm{other}}(t)$	工作区、暂存区和后台占用
$\mathcal{F}(t)\ \mathcal{P}_f$	活动流及流 f的路径
$B_e(t)\ u_f(t)\ w_f(t)$	链路容量、有效吞吐量和剩余传输字节
X_t, U_t, Ξ_t	系统状态、管理操作，以及干扰或新观察到的智能体结- 扰动或新观察到的代理结果
H_t, Π_{adm}	可用历史与可容许非预见策略
$\mathcal{R}, N$	已提交请求队列及其大小
C_i^π, Y_i^π	协议有效客户端交付时间与按时指示函数
$Z^\pi,\ J(\pi)$	队列任务成功率及其期望
$S_k,\ \sigma_k$	潜在状态和观测状态表示
$\widehat{p}_\phi, \phi$	决策时刻 k 学习得到的事件预测器及其参数
$_{\neg\Pi,\, J_k(\Pi)}$	候选反馈计划及其预计的按时交付奖励
$\mathcal{I}_k, \widehat{C}_{i,m}^{\Pi}$	按时结果未确定的请求及展开中的有效交付时间 m
B, K	候选生成尝试次数和已检查动作元组的上限
H, M, L	已检查的动作元组 优化前缀步骤、每个规划的展开次数以及续接事件上限
$\Omega_k,\ \mu_{\mathrm{base}}$	规划时间预算和延续/回退策略

3.1 智能体执行模型

图 1 展示了用户控制的基础设施上由链路$\mathcal{E}$ 连接的异构节点 $\mathcal{N}$ 。工作节点提供推理、隔离工具执行和请求状态存储。逻辑上集中的可靠性控制器利用观测到的工作流、节点/链路状态、资源占用情况以及已确认的检查点来管理客户端请求。

对于请求 i的子任务 v， $\mathcal{C}_{iv}(t) \subseteq \mathcal{N}$ 包含支持其模型、工具、状态表示、运行时版本和任务隔离权限的目标节点。其权威执行器为 $p_{iv}(t)$，若未分配或已暂停，则为 $\perp$。

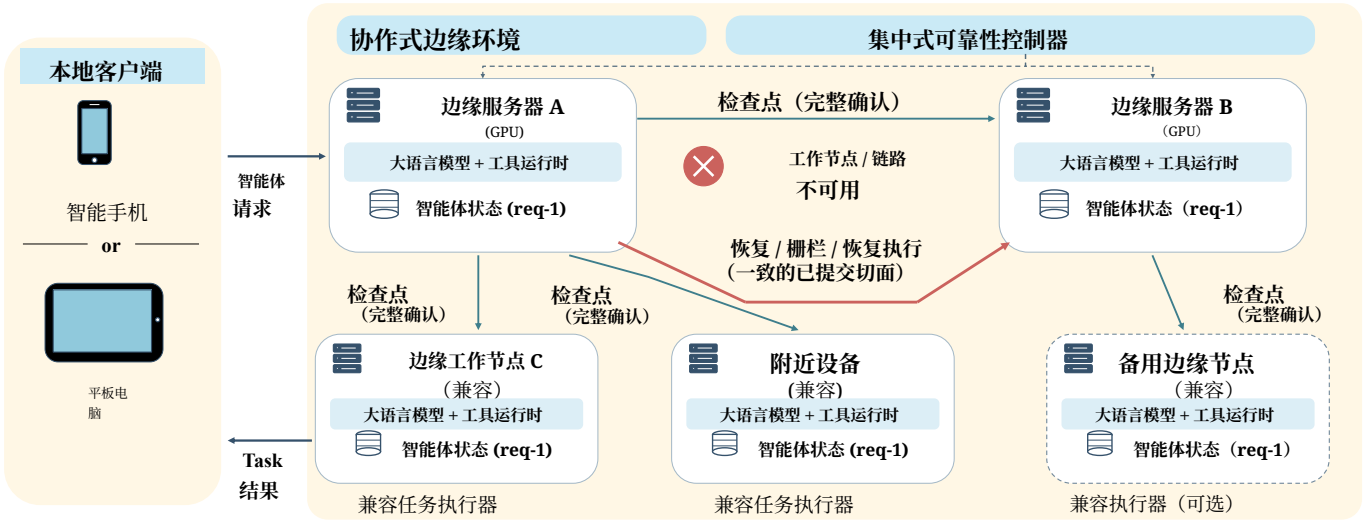


图1. AI智能体执行的协作式边缘架构。

模型和工具均已预先配置。工作节点发生故障或数据链路中断时，控制平面仍然可用，并保留元数据和已提交记录，同时单独跟踪被引用对象。决策依据可能存在延迟的观察；控制延迟和历史记录访问延迟均计入执行时间或恢复时间。

推理和工具交互会揭示后续工作，并生成其所需的输入。请求 i 所揭示的工作流是一个动态有向无环图 $\mathcal{A}_i(t) = (\mathcal{V}_i(t)\ \mathcal{E}_i(t))$，其顶点是推理或工具子任务，其边编码了优先顺序和数据依赖关系。执行结果会改变剩余工作负载 $W_i(t)$；重复操作会创建新的实例，同时保持无环性。只有当子任务所需的前序结果已提交且可访问时，该子任务才会执行。

令 $n_i(t)$ 表示在运行时定义的一致性边界处提交的事件数量。已提交记录标识了工作流版本、推理进度、工具执行结果，以及被引用的制品和效果。后续工作依赖这些历史记录，但被引用对象在发生中断后可能变得不可用。因此，已提交进度不同于可用于恢复的状态。

请求 i 到达 a_i，携带上下文 C_i^{ctx}、模型/工具需求 Ω_i、初始工作负载 $W_i(a_i)$ 以及外部停止规则。其截止时间为 $D_i = a_i + d_i$，其中 d_i 是目标端到端延迟，包括排队、推理、工具、通信和竞争所产生的延迟。各策略共享完成、终止以及推理/工具使用限制，并且只能观察已揭示的工作。

执行既需要可用的计算资源，也需要可访问或能够一致重构的状态。保护进度可以增加恢复选择，但会消耗进行中的工作所需的资源。较新的远程切面可能减少重新计算，却比较旧的本地切面需要更多传输。这些权衡使执行资源、状态保护和恢复选择在剩余截止期限预算下相互关联。

3.2 状态保护与可行恢复

如图2所示，检查点保存了推理上下文、工作流状态、工具结果、文件以及环境/版本信息的一致切面。它不包括预置权重，并且可能包含可导出的 KV 状态；本地句柄和外部影响则需要受支持的重构或协调。

设 $\mathcal{O}_i(k)$ 为在已提交切面 k恢复请求 i 所需的不同带版本对象清单。其逻辑负载（字节数）为

$$V_i(k) = \sum_{q \in \mathcal{O}_i(k)} \mathrm{size}(q). \tag{1}$$

这里，$\mathrm{size}(q)$ 表示对象 q 的大小（字节数）。版本化标识可防止将不兼容状态组合在一起。更新会使 $V_i(k)$具有历史依赖性，且可能呈非单调变化。

设 $k_{in}^{\mathrm{bk}}(t)$ 为节点 n保留的最大完整兼容已确认切点。若无检查点，零仅为记账哨兵，不能证明初始上下文可行重建。其已提交事件滞后为

$$\begin{aligned} \delta_{in}(t) &= n_i(t) - k_{in}^{\mathrm{bk}}(t), \\ 0 &\le k_{in}^{\mathrm{bk}}(t) \le n_i(t). \end{aligned} \tag{2}$$

方程 (2) 衡量的是已提交事件滞后，而非缺失字节数或恢复时间。目录会跟踪完整版本、对象位置、存储层级、进行中的操作和预留容量，包括较早的完整切点。

¯保护操作会在开始时冻结目标切面 $k \le n_i(t)$。若 $\mathcal{B}_{in}(t)$ 包含已验证的兼容目标对象，其增量逻辑负载为

$$\Delta V_{in} = \sum_{q \in \mathcal{O}_i(\bar{k}) \setminus \mathcal{B}_{in}(t)} \mathrm{size}(q). \tag{3}$$

只有完整的清单和对象都经过验证并确认后，检查点才可用。在此之前，部分字节会消耗资源，而先前已确认的切点仍然有效。冻结切点的源对象会

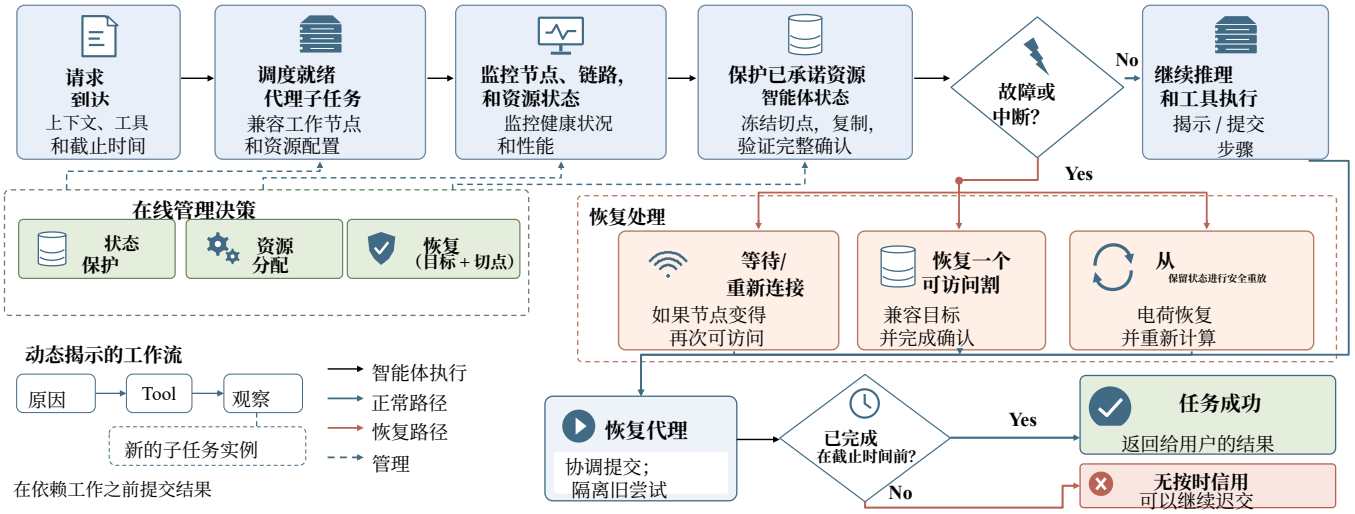


图2。智能体执行、状态保护与恢复。

在传输期间保留，同时后续事件仍可提交。声明的每请求复制上限限制了保留副本的数量。

失败会以两种方式改变状态可用性：崩溃可能破坏易失对象，而通信中断可能使完好对象暂时无法访问。在兼容目标 n 处使用 $\mathcal{K}_{in}^{\mathrm{avail}}(t)$ 进行恢复，即使用本地可用或可通过可行传输获得的完整兼容切面集合。所选切面满足

$$\kappa_i(t) \in \mathcal{K}_{in}^{\mathrm{avail}}(t) \cup \{0\}, \tag{4}$$

资源、工件可用性、版本和重放条件约束。远程切面会产生传输和准备成本；本地可访问割支持独立于主链路的恢复。初始重构，$\kappa_i(t) = 0$，需要保留能够支持安全重放的输入和效果。

恢复会还原所选一致切面，协调持久化结果，并在执行恢复前隔离过时尝试。即使推理输出发生变化，从较早切面的重放或重新计算也必须遵守已提交工具结果和工件版本。外部影响需要操作标识以及受支持的幂等性、结果查询或补偿；不确定的不可逆效果必须在重放前解决。恢复时间包括检测、控制器延迟、排队、传输、恢复、重放、协调和恢复执行，同时计入重叠操作及进一步中断的影响。

3.3 联合管理与资源约束

在每个事件驱动时期，控制器会联合选择部署支持的执行操作、计算资源/带宽预留、检查点创建、刷新、保留或移除操作，以及恢复目标节点和切点。

正常执行、状态保护、重新路由和恢复生成流集合 $\mathcal{F}(t)$。对于每条链路或共享竞争域 e，令 $B_e(t)$ 表示其以字节每秒计的可用容量。每个流 $f \in \mathcal{F}(t)$ 沿路径 $\mathcal{P}_f$ 传输，并获得以字节每秒计的有效吞吐量 $u_f(t)$，满足

$$\sum_{f:e\in\mathcal{P}_f} u_f(t) \le B_e(t), \qquad u_f(t) \ge 0. \tag{5}$$

该约束限制共享e的流的总有效吞吐量，同时考虑执行与状态传输之间的竞争。所需端点或路径不可用时，将无法取得传输进度。故障和连通性可能随时间变化并存在相关性；风险估计覆盖声明时域，但不会透露未来的故障。

在节点 n 上，计算池 r 的可用容量为 $G_n^r(t)$。设 $\gamma_{ivn}^r(t)$ 为请求 i 的子任务 v 的分配，$\gamma^{r,\mathrm{mgmt}}_{,n}(t)$ 为同一池中的保护与恢复总需求。当 n 是请求 i 的子任务 v 的权威活动执行器时，二元变量 $x_{ivn}(t)$ 取值为 1，否则取值为 0。适用时，CPU 池与加速器池彼此独立。可行分配需满足

$$\begin{aligned} \sum_{i,v} \gamma_{ivn}^r(t) + \gamma_n^{r,\mathrm{mgmt}}(t) &\le G_n^r(t), \\ 0 \le \gamma_{ivn}^r(t) &\le x_{ivn}(t) G_n^r(t), \end{aligned} \tag{6}$$

第一个不等式将任务执行以及保护/恢复计入同一计算预算。第二个不等式规定，只有在其权威活动执行器上才能分配任务。子任务暂停时，保护与恢复需求仍计入其中。经性能分析的服务速率取决于任务类型、分配和竞争；不假设其与分配呈线性扩展。

对于节点 n 上的存储层级 h，令 M_{nh} 表示容量，W_{nh} 表示常驻模型权重占用的空间。令 $\mathcal{Q}_{nh}(t)$ 表示已分配的不同状态对象集合，其中 $\mathrm{size}_h(q)$ 表示层级 h 中对象 q 所占用的存储空间。术语 $M_{nh}^{\mathrm{other}}(t)$ 涵盖工作区、暂存区和后台占用。存储可行性要求

$$W_{nh} + \sum_{q\in\mathcal{Q}_{nh}(t)} \mathrm{size}_h(q) + M_{nh}^{\mathrm{other}}(t) \le M_{nh}. \tag{7}$$

该约束对每项物理分配仅计数一次，包括主对象/备份对象、重叠的源端/目标端驻留以及部分传输。使用时，GPU内存、主机内存和工作区存储属于独立层级。剩余预算会扣除现有占用。外生状态或容量损失可能

会暂停或导致工作失败；它不会创建空闲资源，也不会保留已销毁的对象。

分配和兼容性要求

$$\sum_{n\in\mathcal{N}} x_{ivn}(t) \le 1, \qquad x_{ivn}(t) \le \mathbf{1}\{n \in \mathcal{C}_{iv}(t)\}. \tag{8}$$

方程（8）可防止多个或不兼容的权威执行器。新角色还要求已提交且可访问的前置节点、一致状态以及实际的资源预留。重新路由仅在确认目的地就绪状态且旧权威已被隔离后才会改变 p_{iv}（t）；恢复同样要求成功完成重建。重复尝试不能同时提交权威子任务结果。

动作会启动工作，而不是瞬时完成工作。对于剩余传输字节数 w_f（t），

$$w_f(t+\Delta) = \left[w_f(t) - \int_t^{t+\Delta} u_f(\tau)\, d\tau \right]_+ . \tag{9}$$

其中 $[z]_+ = \max\{0, z\}$。方程（5）和（9）根据共享容量计收传输服务费用。已确认的副本只有在其存储层级和故障类型允许的情况下才能继续存在。中断的操作会一直保留已占用资源，直到完成或明确释放。

令 X_t 收集已揭示工作流、执行位置、已提交进度、检查点/对象可用性、队列、进行中的工作、资源占用以及节点/链路状态。对于管理操作 U_t 以及扰动或新观察到的智能体结果 Ξ_t，

$$X_{t+\Delta} = F(X_t, U_t, \Xi_t). \tag{10}$$

转换 F 表示通用的物理执行与核算规则。动作会改变队列、竞争和可恢复性；更改备份放置不会改变外生故障实现。

3.4 问题表述

考虑一个外部生成的队列 $\mathcal{R}$，其中包含 $N > 0$ 个已提交请求，每个请求仅在到达时可见。所有策略使用相同的队列和停止规则。被拒绝、失败、取消和迟交的请求仍计入分母。令 C_i^π 表示策略 π 下完整且协议一致输出的绝对客户端交付时间；若未交付，则为 $+\infty$。一致性涵盖工作流延续、检查点和安全重放。定义 $Y_i^\pi = \mathbf{1}\{C_i^\pi \le D_i\}$ 和 $Z^\pi = N^{-1}\sum_{i\in\mathcal{R}} Y_i^\pi$。

软截止时间决定是否按时成功。迟交请求可能会继续消耗计算资源、存储资源和带宽，直到完成、已观测取消或显式终止触发清理规则。已观测取消会触发用户停止规则，而非由故障驱动的恢复。因此，评分窗口与实际资源生命周期彼此不同。

观测覆盖每个截止时间或更早出现的终止结果，从而确定每个按时完成的结果。

令 H_t 表示可用的观测/动作历史，Π_{adm} 表示资源可行执行器所接受的非预见策略。我们寻求一种可接受在线策略，使其联合选择执行资源、状态保护操作以及可行的恢复目标和切点，以最大化任务成功率的期望值：

$$\underset{\pi\in\Pi_{\mathrm{adm}}}{\text{maximize}} \quad J(\pi) = \mathbb{E}[Z^\pi] = \frac{1}{N}\sum_{i\in\mathcal{R}} \Pr(C_i^\pi \le D_i), \tag{11}$$

满足方程 (5)–(7) 和 (8)，以及依赖关系、检查点一致性、可访问切点、安全重放和权限转换条件。策略选择 $U_t \sim \pi(\cdot \mid H_t)$，执行遵循方程 (10)。

期望涵盖请求特征、推理/工具执行结果、失败、网络变化和策略随机化。对于固定规模的队列，无需假设请求结果相互独立，方程 (11) 中的等式仍然成立。状态可用性和完成时间是联合评估的，而不是通过边际概率相乘得出。恢复中断、延迟、传输字节数和存储开销是补充指标，而非取代方程 (11) 的任意惩罚项。第 4 节将为此问题提出一种基于世界模型的模型预测控制方法。

4 基于世界模型的模型预测控制

我们提出基于世界模型的模型预测控制（WorldMPC），通过联合建模工作流进度和可恢复状态，实现可靠的智能体执行。该模型生成遵循已提交历史的执行和恢复轨迹。这些轨迹为联合资源、保护和恢复决策提供指导，以实现第 3 节中的任务成功目标。

4.1 部分可观测半马尔可夫决策过程

我们将在线执行管理建模为部分可观测半马尔可夫决策过程 (POSMDP) [34],[35], ，其中执行状态只能被部分观测，且决策间隔可变。

潜在状态与决策时刻。在决策时刻 k，潜在状态为 $S_k = (t_k,\ X_{t_k},\ \eta_k)$，其中 t_k 为物理时间，X_{t_k} 为执行状态，η_k 则包含马尔可夫性质所需的其他信息，例如服务时长、故障模式和待处理通知。该状态还会跟踪按时交付状态。我们假设在管理操作下，增广状态满足马尔可夫性质。

决策由请求到达、子任务完成、检查点确认、检测到中断或周期性控制定时器触发。决策时刻之间的间隔 $\Delta t_k = t_{k+1} - t_k$ 会随时期变化。半马尔可夫转移核为

$$Q(ds', d\delta \mid s, u) = \Pr\{S_{k+1} \in ds', \ \Delta t_k \in d\delta \mid S_k = s, U_k = u\}. \tag{12}$$

这里，s和 s' 分别表示当前和下一个潜在状态，u表示动作，δ 表示经过的时间。

动作与目标。为将 POSMDP 奖励与任务成功率目标联系起来，考虑一个固定的请求评估队列 $\mathcal{R}$。令 $\ell_i(s) \in \{0, 1\}$ 表示

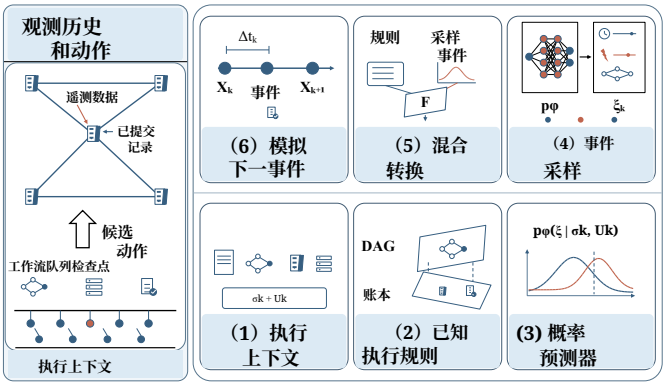


图 3. 用于 AI智能体执行的混合世界模型。

请求 i 在状态 s 下是否已实现有效按时交付。该指示函数是非递减的。阶段奖励为

$$r(s,u,s',\delta)=\frac{1}{N}\sum_{i\in\mathcal{R}}[\ell_i(s')-\ell_i(s)]. \tag{13}$$

将从初始状态开始、直至所有按时结果均已确定期间各次转换的奖励相加，即可得到任务成功率。评估队列是固定的，但在线控制器无法预知未来的请求。

在线决策规则。 令 H_{t_k} 包含观测到的遥测数据、事件、已提交执行记录和先前动作。控制器利用这些历史记录，在不计算精确 POSMDP 信念的情况下选择可行操作：

$$\begin{aligned}\sigma_k &= \mathrm{Enc}(H_{t_k}),\\ U_k &= \mu_\phi(\sigma_k)\in\mathcal{U}_{\mathrm{feas}}(H_{t_k}),\end{aligned} \tag{14}$$

这里，Enc 构建观测状态 σ_k，而 $\mathcal{U}_{\mathrm{feas}}$ 是可行动作集。WorldMPC 使用世界模型参数 ϕ实现 μ_ϕ，无需单独训练策略。符号 U_k 是 U_{t_k} 在系统模型中的简写。

4.2 世界模型

我们的混合世界模型刻画了工作流进度、可恢复状态和共享资源的耦合演变。执行会产生新状态，保护会消耗资源以维持该状态，而中断会改变仍可用于恢复的状态。我们采用已知执行规则和学习得到的概率动力学来建模这些效果，并以离散事件系统建模 [36]和概率动力学建模 [37]为基础。图 3 展示了观测历史和候选动作如何定义执行上下文（1）。已知规则用于管理工作流依赖关系、资源核算和已提交效果（2），而概率预测器则对执行时间、中断和工作流演变进行建模（3）。从预测器中采样事件变量（4），并由混合转移依据已知规则进行处理（5），从而生成下一个模拟事件、经过时间、执行进度和可恢复状态（6）。

已知执行规则。规则组件实现了第3.3节中的转换与记账机制 F。它会更新任务依赖关系、角色所有权、资源预留、传输进度和已确认检查点。可用的恢复切点需要完整的清单和对象、一致的已提交效果，以及可行的恢复或重放路径。

概率转移模型。 让 ξ_k 收集推进执行所需的不确定量，包括剩余服务时间、节点和链路变化、恢复成本，以及新揭示的工作流信息。学习组件进行采样

$$\begin{aligned}\widehat{\xi}_k &\sim \widehat{p}_\phi(\cdot\mid\sigma_k,U_k),\\ (\widehat{X}_{k+1},\widehat{\Delta t}_k) &= \mathcal{F}(\widehat{X}_k,U_k,\widehat{\xi}_k),\end{aligned} \tag{15}$$

其中， $\mathcal{F}$ 是 F的事件驱动实现。在此，$\widehat{\ }$ $\widehat{\ }$ $\sim X_k$ 和 X_{k+1} 是模拟执行状态， Δt_k 是预测的下一事件发生时间，$\widehat{\ }$是 ξ_k，其中包含采样得到的不确定事件变量。它将执行、传输和资源占用推进至下一事件，同时由控制定时器限定决策之间的时间间隔。观察结果和已确认记录初始化仿真。

预测器特征描述子任务和工具类型、模型和设备标识符、资源分配、已用服务时间、队列、检查点负载及其存留时间，以及近期的网络和故障观测。根据已用服务时间进行条件化，可以支持对正在运行的操作进行剩余时长预测。共享历史或场景变量可以表示相关中断。工作流预测描述受支持的依赖扩展以及工作负载或状态大小变化；实际工具输出和已提交文件通过执行进入系统。

模型训练。 每条记录 $\zeta_j=(\sigma_j,\ U_j)$ 都使用观测到的事件目标 y_j之前可用的历史信息。目标包括事件时间、事件类型，以及观测到的工作流、资源或恢复变化。似然使用与未来采样相同的事件模型，并通过掩码和边缘化处理不可用字段。训练集和验证集划分会将共享拥塞和中断的请求放在一起。

对于完全观测的事件目标和行政删失事件区间，训练目标分别为

$$\begin{aligned}\mathcal{L}(\phi)=-\frac{1}{|\mathcal{D}|}\Bigg[&\sum_{j\in\mathcal{D}_{\mathrm{obs}}}\log\widehat{p}_\phi(y_j\mid\zeta_j)\\ &+\sum_{j\in\mathcal{D}_{\mathrm{cens}}}\log\widehat{S}_\phi(c_j\mid\zeta_j)\Bigg],\end{aligned} \tag{16}$$

这里， $\mathcal{D}_{\mathrm{obs}}$ 和 $\mathcal{D}_{\mathrm{cens}}$ 分别包含观测到的事件记录和删失事件记录， $\mathcal{D}$ 是完整训练集。对于每条删失记录 j， c_j 是观测到的无事件区间， $S_\phi(c_j\mid\zeta_j)$是该区间内未发生事件的预测概率。完成和中断是不同的事件类型。崩溃会监督中断目标，而不完整记录则贡献其观测字段或有依据的删失项。每个目标只计入一次。对于每个训练小批次，参数按以下方式更新：

$$\phi\leftarrow\phi-\eta_\phi\nabla_\phi\mathcal{L}(\phi), \tag{17}$$

其中 η_ϕ 为学习率。我们使用针对事件时间似然的明确验证标准来选择模型，并

算法1 混合世界模型的监督训练

要求： 执行轨迹 $\mathcal{D}$、训练轮次 E、学习率率 η_ϕ
确保： 训练后的模型 $\widehat{p}_\phi$ 和固定的特征定义
1: 构建历史条件记录和观察到的目标掩码
2: 将执行运行划分为训练集和验证集
3: 初始化概率预测器参数 ϕ
4: **对于**时期 $= 1$至 E **执行**
5: **对于** 每个训练小批次**执行**
6: 评估观察似然和删失似然在方程 (16) 中
7: 更新 $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}(\phi)$
8: **结束循环**
9: 评估预测似然和中断校准在验证运行中
10: 保留声明的最佳模型版本验证标准
11: **结束循环**
12: 冻结选定的模型版本以供在线规划
13: **返回** $\widehat{p}_\phi$ 和特征定义

在不同工作负载和故障条件下进行中断校准。所选模型在每次规划调用期间保持不变；基于轨迹的更新在调用之间生效。

在评估的实现中，持续时间分布和事件概率或速率根据观察进行拟合，并为未测量过程设定声明的先验分布。第5节将详细说明这些组件。

算法1总结了世界模型训练。第1–3行构建历史条件记录，划分训练运行和验证运行，并初始化预测器。第4–8行使用方程（16）中的观察到的和删失事件目标拟合预测器。第9–12行使用预测似然和中断校准选择并冻结规划模型。

4.3 状态与动作空间

我们将控制器在时期 k 的观测状态表示定义为

$$\sigma_k = (g_k, d_k, e_k, q_k, c_k), \tag{18}$$

此处， g_k 表示已揭示工作流与执行进度， d_k 跟踪剩余截止时间， e_k 描述观测到的设备与网络条件， q_k 记录执行与传输队列， c_k 表示检查点可用性与恢复状态。

在每个决策时刻，控制器选择一个动作，或一批联合可行的动作。该动作定义为

$$U_k = (u_k, \vartheta_k, z_k, \rho_k), \tag{19}$$

其中， u_k 表示动作类型， ϑ_k 表示请求、子任务与目标， z_k 表示检查点操作与目的地， ρ_k 表示计算资源/带宽预留。

执行动作类型和参数域由部署场景定义。在每个决策时期，观察到的工作流状态、资源可用性和恢复条件决定哪些动作可行。

并发动作必须满足资源总量、兼容性和恢复约束。当没有可行动作时，执行器会等待下一事件或控制定时器，同时已接受的工作继续进行。

4.4 奖励估计

规划器估计当前观察到的请求的预期准时交付奖励。令 $\mathcal{I}_k$ 表示在时期 k时其按时结果仍未确定的请求集。对于候选反馈计划 Π，令 $\widehat{C}_{i,\,m}^{\Pi}$ 表示在世界模型推演 m中请求 i 的有效客户端交付时间；若未发生有效交付，则表示为 $+\infty$。使用 M 次完整推演，估计奖励为

$$\widehat{J}_k(\Pi) = \frac{1}{M|\mathcal{I}_k|} \sum_{m=1}^{M} \sum_{i \in \mathcal{I}_k} \mathbf{1}\{\widehat{C}_{i,m}^{\Pi} \le D_i\}. \tag{20}$$

有效交付要求工作流一致性和安全恢复。所有候选计划使用相同的请求集 $\mathcal{I}_k$，因此不同计划之间的估计奖励具有可比性。该评分是条件规划估计，而报告的任务成功率涵盖所有已提交请求。若出现平局，则根据预测的恢复中断、传输字节数和固定候选顺序重新解决。新到达的请求会触发重新规划；空的 $\mathcal{I}_k$ 会调用可行的延续策略。

4.5 基于世界模型的模型预测控制

图 4 展示了 WorldMPC 如何结合模型训练、预测规划和执行反馈，以联合选择资源分配、状态保护和恢复操作。历史轨迹用于训练世界模型的预测器，而在线观测构成控制器状态。规划器在 M个采样未来状态上评估反馈计划和全回退计划。反馈计划使用一个 H步的优化前缀，并在需要时采用通用延续策略，其评分基于有效按时交付。

可行候选生成。 控制器根据部署支持的动作类型和参数域构建$\mathcal{L}_K(\sigma)$ ，并结合符合条件的子任务、兼容目标、保留的恢复切点、资源配置和检查点操作。在执行期间，保护和资源调整可能仍然可行。元组按照请求截止时间、请求标识符、目标标识符、资源配置标识符和保护选项排序。在剩余时间预算内，生成器以惰性方式检查至多 K 个元组，并保留可行动作；任何可行的回退也会单独保留。该排序规定的是枚举顺序，而不是任务成功目标。

除全回退计划外，每个候选项都是一个排名序列 $\boldsymbol{r}^b = (r_0^b, \ldots, r_{H-1}^b)$，其中有 $r_h^b \in \{1, \ldots, K\}$。在 H 个模拟决策步骤中的每一步，它都会根据轨迹历史重新构建选项列表，并应用请求的排名；如果该排名不可用，则进行回退。因此，该计划能够响应已揭示的完成情况和失败。生成过程包括全回退计划；在上限允许的范围内覆盖不同的可行首个动作，并对其余候选项使用带种子的排名序列采样。最多进行 B 次生成尝试，并删除重复项。

蒙特卡洛展开。每个候选方案都会在从方程（15）采样的M个联合执行场景中进行评估。每条

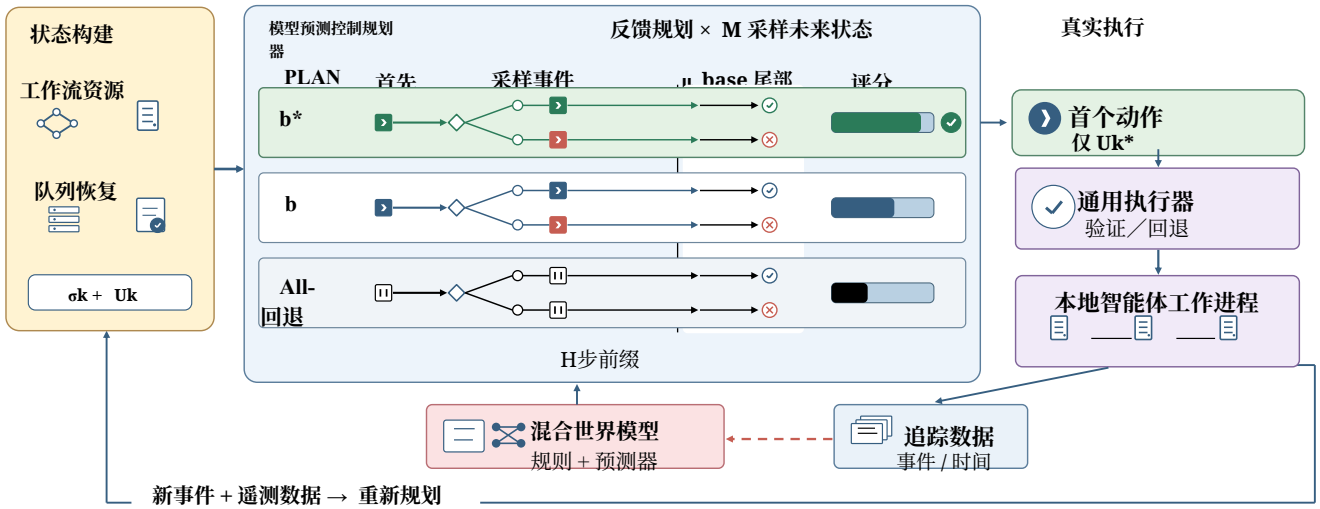


图 4. WorldMPC框架。

轨迹都会遵循任务执行、状态保护和可能的恢复，同时与已提交的推理输出、工具结果及文件保持一致。它还会考虑共享资源争用，以及沿可行恢复路径所需的传输、恢复和重放。每个模拟决策都会重新验证。候选评估使用公共随机种子，但不同动作可能产生不同的事件序列。

在优化后的 H 步前缀之后，每条轨迹都使用相同的延续策略 μ_{base} 来实现完整请求交付。只有当 $\mathcal{I}_k$ 中的每个请求都已交付或错过其按时目标时，才能获得完整评分，最迟不超过 $T_k^{\text{score}} = \text{最大值}_{i\in\mathcal{I}_k} D_i$。逐请求账本会记录每次交付。

延续策略会保留正在运行的工作和已确认的检查点。在受支持的调度边界处，被中断的请求和就绪请求遵循最早截止期限优先原则，并以请求年龄和标识符打破平局。对于被中断的工作，它会选择一个兼容目标和切点，使预测恢复时间加剩余执行时间最小；对于就绪工作，它会在相同的资源配置上，使预测可行完成时间最小。不可行的工作会等待，同时其他可行工作继续执行。当可行的按时完成工作无法使用资源时，迟到的就绪工作才会得到服务，因此持续过载可能会推迟其执行。

有界规划和运行时安全机制。上限B、 M、 H、 L 以及预算 Ω_k 共同限制规划范围。只有当所有 M 评分展开都能在这些限制和受支持的上下文内完成时，候选方案才具备资格。排除候选方案不会将真实请求标记为失败；如果没有候选方案符合条件，控制器将使用 μ_{base}。迟到的工作会保留资源，正在运行的工具仍不可抢占，已观测取消会遵循停止规则。已完成展开、排除和回退计数会跟踪搜索限制和仿真成本。

规划和状态访问会消耗请求时间，同时已接受的任务继续执行。展开会按照当前分配推进现有任务，推进时长为声明的规划延迟估计值。若没有可行操作，则推进至下一事件或计时器触发。执行前，控制器会刷新遥测数据，并验证所选的首个动作，或采用可行的回退方案。如果两者都不可用，已接受的任务会继续执行，直至下一事件或计时器触发。实际规划时间计入执行时钟。

对于有界工作流和选项列表，展开评估的成本为 $O(BM(H+L)c_{\text{step}})$，不包括状态构建和候选生成。每个事件步骤都包括预测器评估、状态更新、选项列表重构和可行性检查。评估后的参数化预测器在控制器CPU上运行，不会产生额外的大语言模型调用；顺序评估仅保留活跃轨迹。所测预算包括生成和评分。实验报告上限、资源配置、验证标准、延迟估计和回退频率。

算法 2实现在线决策循环。第1–7行构建观测状态、初始化回退方案并生成可行计划。第8–27行模拟优化前缀和共同延续部分，计入规划延迟，并仅对具有 M 完整展开的候选项进行评分。第28–31行选择评分最高的计划；如果没有计划符合条件，则保留回退方案。第32–36行刷新观察结果，验证并执行首个动作或可行的回退动作，记录结果和开销，然后等待下一事件或控制定时器。

5 性能评估

我们使用第3.4节中的任务成功率（TSR）评估 WorldMPC：所有已提交请求中，其完整且协议一致的输出按时到达客户端的请求所占的比例。

5.1 实验设置

5.1.1 数据集

该公共数据集为 SWE-bench Verified，其中包含来自500个问题任务和12个代码仓库 [38]。我们将问题文本长度映射到工作流深度、名义操作工作量和执行状态大小，从而构建模拟工作流。这些工作流交替进行推理和工具阶段，并以回执结束。

该合成数据集均衡地涵盖了纯推理、只读工具和读写工具工作流。每个请求包含三个至五个按顺序逐步揭示的阶段，每次执行一个操作，其中

算法 2面向可靠智能体的基于世界模型的模型预测控制执行

要求： $\widehat{p}_{\phi}$、μ_{base}，容量 B，K，H，M，L，预算 Ω_k

确保：可行执行、保护和恢复

当服务处于活动状态**时执行**
  观察事件、遥测数据、确认应答和取消
  构造 σ_k 和固定请求集 $\mathcal{I}_k$
  选择全回退计划
  如果 $\mathcal{I}_k \neq \emptyset$，动作可行且上下文受支持**则**
    启动计时器；设置 $\mathcal{P}_k = \emptyset$
    使用 $\mathcal{L}_K$ 和 H生成 $\leq B$ 个计划
    对于每个规划 Π^b 只要预算尚有剩余**则**
      初始化展开结果
      对于 $m = 1$至 M 只要预算尚有剩余**则**
        根据观察初始化状态/账本
        按估计的规划延迟推进
        运行 Π^b，直到解决或发生 H 个事件
        如果评分结果未解决 **则**
          对 μ_{base} 个事件执行
        结束如果
        如果展开未完成或不受支持**则**
          丢弃 Π^b
          中断 展开循环
        else
          每次有效交付仅记录一次
        结束 if
      结束 for
      如果所有 M 展开均在预算内完成 **则**
        根据方程 (20) 评分；将 Π^b 添加到 $\mathcal{P}_k$
      结束 if
    结束循环
    如果 $\mathcal{P}_k \neq \emptyset$**则**
      选择评分最高者，使用已声明的平局判定规则
    结束条件判断
  结束条件判断
  刷新观察；仅编译首个动作
  执行有效动作或可行的回退；否则继续当前执行
  记录规划时间、动作、事件和结果
  等待事件或计时器，同时继续已接受的工作
结束 while 循环

每次执行一个操作；读写请求以回执结束。操作工作量具有乘性的对数正态变异，完成操作后状态大小会增加。请求按照间隔或突发模式到达。相对截止时间为 $d_i = \alpha$（$1.5W_i + 12$）秒，其中 W_i 为总名义工作量，截止时间因子 α 对参考截止时间（$1.5W_i + 12$）秒进行缩放。

5.1.2 仿真环境

仿真中有两个异构工作节点，共享一条100 Mbps 链路。两者的最快与最慢服务速率之比为二，总名义计算能力归一化为两个参考工作器单位。每个工作节点都提供独立的推理池和工具池，以及384 MB 的状态内存。执行、检查点准备、恢复和数据传输均遵守由通用执行器强制执行的共享资源约束。正在运行的操作不可抢占。可行且未完成的按时工作优先于迟到的就绪工作，而迟交执行仍会占用资源。

表 2 主要仿真与规划参数。

参数	值
每个周期的请求数	18，27，36；传输：36，45
独立评估种子	24; 4
虚拟工作者	2，4，8; 传输：4，6
共享带宽（Mbps）	25，100，400；传输：60
最快/最慢服务速率比	$1, 2, 4$
每个工作进程的状态内存（MB）	384
截止时间因子 α	1.5、3、4.25、6
失效率因子	0,1, 2
每个请求已确认的远程副本数	最多1
基线保护间隔（秒）	8
控制制定时器间隔（秒）	2
候选项上限 B / 选项上限 K	$4/8$
前缀深度 H / 样本 M	$2/4$
续接事件上限 L	300
规划预算（毫秒）	20
每请求重试上限	12 次尝试/操作
回合清理时限（秒）	900

WorldMPC 从 $\{1, 0.5\}$ 中选择计算资源分配比例，并可选择保护动作；四种基线方法均使用比例一，并在支持的已提交边界处每8秒进行一次周期性检查点保存。服务速率随分配比例的0.8次幂缩放。所有方法共用依赖关系检查、兼容性约束、传输计量和恢复规则。因此，该比较评估的是完整的控制策略，包括其分配和保护选项，而非仅单独考察预测规划。

五种环境考察了预测难度的不同来源：健康运行、可观测压力、不透明故障、共因中断和误导性压力。它们在工作节点故障风险、压力信号的信息量以及相关中断方面各不相同。非健康环境还包括共享链路中断。这些条件是预先设定的，并非实测的物理故障分布。

仿真中的学习得到的事件预测器使用八个拟合标量：推理、工具和回执时长预测器各自的对数位置参数和对数尺度参数，以及两个压力条件下的中断率。时长预测器使用删失对数正态似然，以100 个样本进行全批量更新，学习率为 0.01；中断率则使用暴露记录和弱Gamma先验进行估计。训练集和验证集分别包含2400 和800 条合成观测。对于 N 条训练记录和 E 次更新，拟合这些固定大小的预测器需要 $O(EN)$。拟合后的模型在评估期间保持固定。展开使用按工作节点划分的风险率，以及当前链路中断时的恢复先验。

每个数据集均在五种环境和两种到达模式下进行评估，每次运行包含27个请求，并使用24 个配对随机种子。对于每个种子，各方法使用相同的请求、工作节点配置和中断日历。工作节点数量扫描在工作负载固定时增加容量，而异质性扫描则保持总计算资源不变。主要参数范围见表 2。仿真使用 Python 在 Intel Core i5-12600KF 上运行，测得的规划时间计入执行时钟。

5.2 基线方法

我们将 WorldMPC 与四种基线方法进行比较。异构最早完成时间（HEFT）根据当前已揭示有向无环图中基于计算资源和通信估计值计算出的向上秩，对就绪操作进行排序。最早

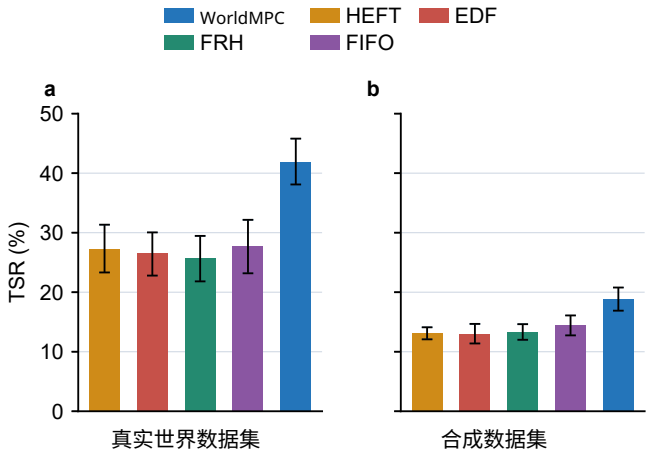


图5. 真实世界数据集和合成数据集上的任务成功率。

截止期限优先（EDF）按照绝对请求截止期限对就绪任务进行排序，并以到达顺序打破平局。先进先出（FIFO）遵循请求到达顺序。失败风险启发式（FRH）遵循到达顺序，并优先选择根据观测压力和失败历史估计出的当前风险最低的放置位置；若仍然相同，则根据预计完成时间打破平局。

5.3 实验结果

5.3.1 世界模型训练与预测

独立验证集用于衡量事件时间预测和中断校准。未删失操作的操作时间负对数似然为 0.0217，平均绝对持续时间比误差为 0.2002。在生成的10–35秒暴露区间内，中断预测的布里尔评分为 0.0538，负对数似然为 0.1966。删失记录会贡献生存似然项。这些数值反映了在合成数据分布下拟合的参数化世界模型的特征。

5.3.2 总体性能

图 5(a) 对比了真实世界数据集上的 TSR，以评估 WorldMPC 的联合执行决策。与 HEFT、EDF、FRH 和 FIFO 相比，WorldMPC 的 TSR 分别提高约54.1%、57.9%、63.3%和51.6%。观察到的提升表明，在选择动作时综合考虑执行时间、资源争用和恢复成本是合理的。

图 5(b) 对比了合成数据集上的 TSR。与 HEFT、EDF、FRH 和 FIFO 相比，WorldMPC 的 TSR 分别提高约44.1%、46.2%、42.1%和30.4%，同时平均交付延迟分别降低约20.3%、11.1%、19.8%和15.8%。这两方面的改进都与在共享资源约束下综合考虑等待、执行和恢复相一致。

5.3.3 工作负载和资源的效果

图 6(a) 评估了固定资源下请求数量对 TSR 的影响。将请求数量从18 增加到36 会降低 TSR，因为更多执行和状态传输操作会争夺计算资源和带宽。在36 个请求下，WorldMPC 的 TSR 分别比 HEFT、EDF、FRH 和 FIFO 高约150。0%、150。0%、42。9%和35。1% 。WorldMPC 根据竞争、恢复工作和剩余截止时间评估候选决策。

图 6(b) 考察了共享带宽对 TSR 的影响，此时执行、保护和恢复传输会竞争同一链路。更高的带宽能够提升 WorldMPC 的 TSR。在400 Mbps 下，其 TSR 分别比 HEFT、EDF、FRH 和 FIFO 高约32。8%、1。1%、25。4%和18。7%。与 EDF 之间观察到的差距较小，这表明当通信压力降低时，按截止时间排序的策略会更具竞争力。

图 6(c) 评估了在请求数量和共享带宽固定的情况下，WorldMPC 如何利用额外工作节点。增加工作者数量可提升 WorldMPC 的 TSR。当工作节点数量为 8 时，与 HEFT、EDF、FRH 和 FIFO 相比，WorldMPC 的 TSR 分别高出约82.4%、82.4%、325.6%和82.4% 。额外工作节点拓展了执行和恢复选择，WorldMPC 会结合资源分配和剩余截止时间对这些选择进行评估。

图7显示，在总容量固定的情况下，对计算资源异构性的响应呈非单调性。在最大的速度比下，与 HEFT、EDF、FRH 和 FIFO 相比，WorldMPC 的 TSR 分别高出约143.5%、133.3%、69.7%和24.4% 。更大的速度差异会产生颇具吸引力的快速工作节点部署，但也可能使竞争集中于这些工作节点上。这种权衡与非单调曲线一致：仅有更大的异构性并不能确保更高的 TSR，而在测试的最大速度比下，协调决策仍然有益。

5.3.4 可靠性与有状态恢复

图8(a)显示，更严重的失败会降低按时完成率。失败会通过等待、传输、恢复和重建消耗剩余预算。在合成数据的主要对比中，与 HEFT、EDF、FRH 和 FIFO 相比，WorldMPC 将平均恢复中断时间分别降低约3.9%、6.6%、11.9%和12.3% 。

由于所有方法共用同一个恢复执行器，观测到的平均值反映了其调度和保护选项所产生的恢复事件。在真实世界数据集上，尽管 WorldMPC 的 TSR 更高，其平均中断时长仍比 HEFT 高6.8% ，比 FRH 高6.8%。由此可见，仅将中断时长降至最低并不等同于最大限度地提高任务及时完成率；评估恢复时必须同时考虑剩余执行工作量和截止时间。

图8(b)显示，TSR 会随着截止时间因子 α 的增加而提高。平均而言，WorldMPC 的 TSR 分别比 HEFT、EDF、FRH 和 FIFO 高47.1%、19.6%、71.3% 和44.8% 。WorldMPC 在剩余截止期限预算内选择动作时，会综合考虑执行时间、资源争用和恢复成本。

5.3.5 未见条件下的泛化分析

这两种组合配置在工作负载和资源条件变化的情况下，评估相同的训练后的模型和规划上限。在第一种配置中，WorldMPC 的表现与 HEFT 和 EDF 相当，比 FRH 高约72.4%，并与 FIFO 相当。在第二种配置中，其 TSR 分别比 HEFT、EDF、FRH 和 FIFO 高约25.0%、25.0%、78.6%和25.0%。在这两种配置下，WorldMPC 无需重新训练，仍具有竞争力或表现更佳。

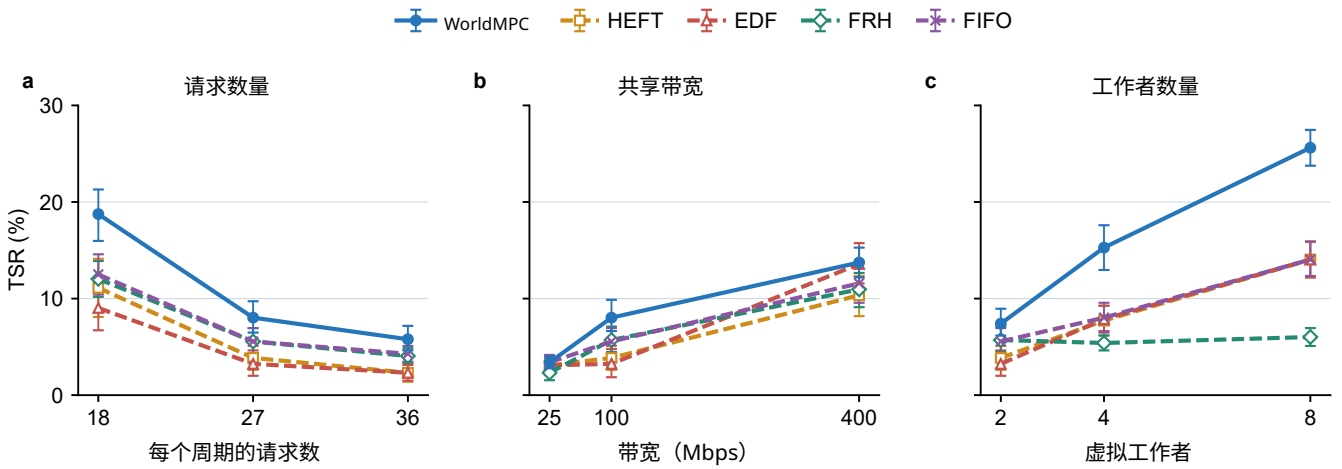


图6. TSR与请求数量、共享带宽和工作节点数量的关系。

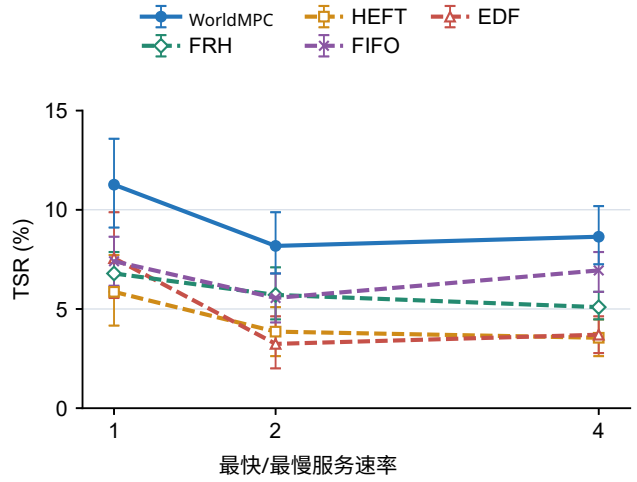


图7. 不同计算异构性水平下的 TSR。

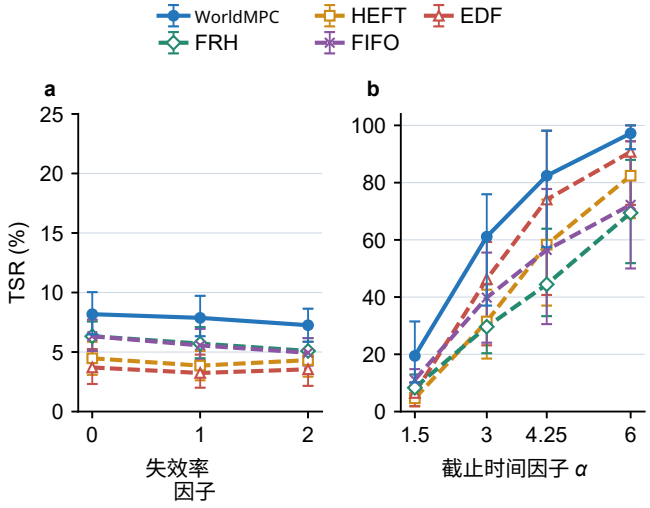


图 8. 不同故障率和截止期限因子下的 TSR。

5.3.6 开销与组件分析

在主要的合成比较中，WorldMPC 每次控制器回调的平均耗时为 11.42 毫秒，86.17% 的蒙特卡洛规划调用会触发回退。回调时间包含未启动搜索的选择操作；回退则使用实际搜索以调用次数为分母。协作式预算检查发生在展开事件之间，因此回调耗时可能超过20 毫秒。检查点流量和共享链路总流量的平均值分别为每个提交请求0.02 MB 和45.76 MB。

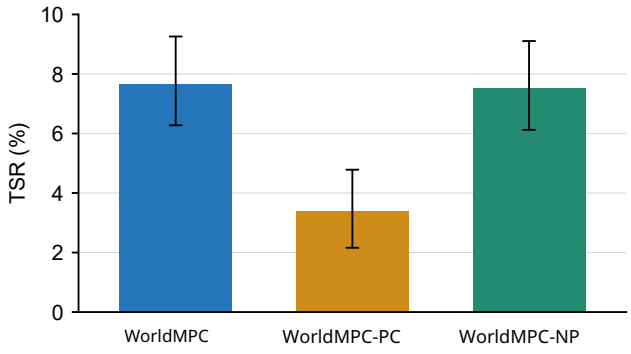


图 9. 不同状态保护机制下的 TSR。

图 9 对比了不同保护机制，同时保留了采样式规划。与 WorldMPC-PC 相比，完整的 WorldMPC 将 TSR 提高约 125.8%，共享链路流量降低39.3% 。这一组合与减少周期性检查点传输相符，因为这些传输会与执行流量和恢复流量竞争。与 WorldMPC-NP 相比，完整的 WorldMPC 将 TSR 提高约2.1%，流量则增加1.8%。WorldMPC-NP 保留了已提交日志重建，因此移除额外检查点并不会移除所有恢复能力。

5.3.7 可观测故障风险下的预测规划

我们进一步评估预测规划对 TSR 的贡献。我们将 WorldMPC 与基线方法及其变体进行比较。此外，我们还考察规划预算如何影响 TSR 和回退。评估中还纳入了规划和恢复成本。

图 10(a) 对比了 WorldMPC 与 HEFT、EDF、FRH 和 FIFO。与各基线相比，WorldMPC 的 TSR 分别提高了 16.7%、21.7%、75.0% 和 21.7%。WorldMPC 会评估执行选择对剩余工作流完成情况的影响。因此，在选择动作以提升 TSR 时，它能够权衡执行速度与中断和恢复成本。

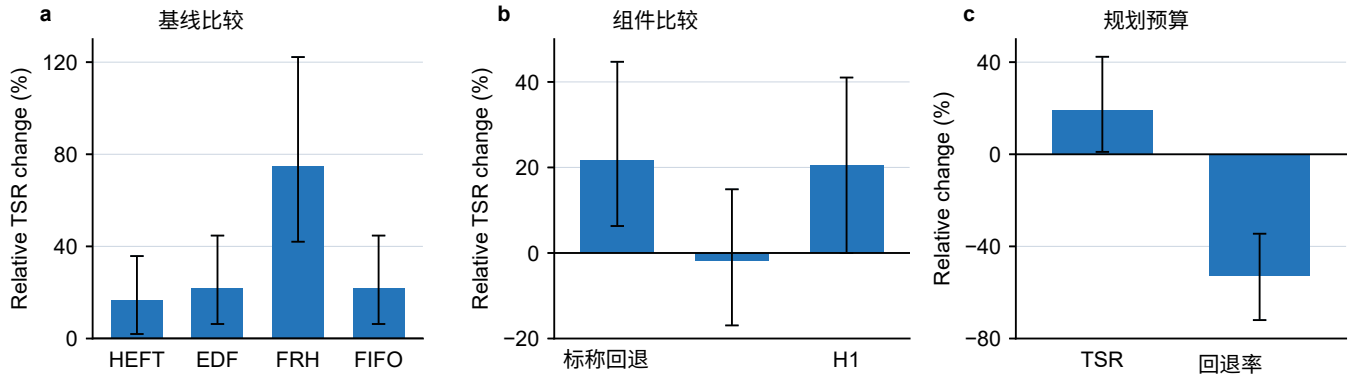


图 10. 可观测故障风险下的预测性规划。

图 10(b) 对比了 WorldMPC 与仅回退执行、固定名义预测器以及单命令前缀。这些变体分别移除了轨迹搜索、将学习型预测器替换为固定先验，或缩短了优化后的动作前缀。与仅回退执行相比，WorldMPC 的 TSR 提高了 21.7%；与单命令前缀相比，则提高了 20.4%。通过评估不同的执行计划，WorldMPC 能够考虑当前动作对后续执行和恢复的影响。

图 10(c) 比较了两种规划预算。与较短的预算相比，较大的预算使 TSR 提高了 19.1%，并将回退率降低了 52.6%。较大的预算允许对更多候选计划进行完整评估，从而为 WorldMPC 提供更多改进 TSR 的机会。

5.4 真实世界测试平台评估

5.4.1 测试平台设置

我们开发了 PolyLink，这是一个用于异构设备 [39]间分布式大语言模型推理的协同边缘平台。它连接了来自数十所大学实验室的 100 多个消费级GPU [6]。此类基础设施可以为 AI智能体提供推理资源，而 AI智能体的多步执行还要求在发生中断时维持进度并保留可恢复状态。我们在 AI智能体执行原型中实现了 WorldMPC，以协调该环境中的资源分配、状态保护和恢复。

可靠性控制器将智能体阶段调度到相互隔离的工具工作区，用于文件读取、推理、算术或问题元数据处理以及结果验证。已提交的结果和验证结果会揭示后续阶段或修复操作。我们在执行期间注入智能体进程终止以及工作节点侧检查点存储丢失。基于检查点的恢复会在恢复智能体上下文和文件之前验证已保存切面。工具阶段从已恢复状态重新执行，控制器则拒绝来自过时尝试的结果。

真实世界测试平台使用两台连接到现有模型服务的 RTX 4090 工作节点主机。每个工作节点一次接纳一个智能体阶段。工具容器使用两种资源配置：0。5 个 CPU、256 MiB 内存，以及1 个 CPU、512 MiB 内存。检查点存储限制为每个工作节点64MiB。导出和导入共享一条串行化控制器路径，传输限制为128 KiB/s，512

KiB/s 和 2MiB/s。控制器端单调时钟用于测量

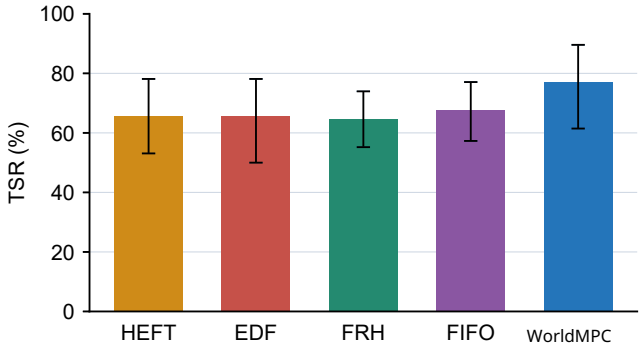


图11。真实世界测试平台上的任务成功率。

完整响应交付情况。TSR 统计在截止时间前交付的协议完整输出。任务正确性则依据预期算术结果或问题元数据单独进行检查。

真实世界测试平台使用一个世界模型，其中包含十二个对数正态时长预测器和十二个协议成功概率，涵盖不同的工作节点、操作类型和资源配置。这些组件根据独立的校准观测进行拟合，以捕捉执行时间和协议完成情况的变化。结合执行规则，它们支持预测候选管理动作下的工作流进度和可恢复状态。

5.4.2 性能比较

图11比较了五种方法在真实世界测试平台上的 TSR。WorldMPC 达到77。08%，而 HEFT、EDF、FRH 和 FIFO 分别为65。63%、65。63%、64。58%和67。71% 。WorldMPC 在选择执行动作时会考虑资源竞争和恢复成本。这有助于让更多请求在截止时间内完成。

6 结论

在本研究中，我们针对节点故障和无线中断条件下协同边缘计算中的可靠性感知型 AI智能体执行问题展开研究。我们提出了一种基于世界模型的模型预测控制方法，用于联合优化资源分配、状态保护和恢复，以应对具有相互依赖推理和工具进度且不断演化的智能体工作流，并满足任务

截止期限。实验结果表明，与所评估的基线方法相比，所提出的方法提高了任务成功率。本研究有助于在动态边缘环境中可靠地完成有状态智能体任务。未来的研究将重点关注该方法在多样化智能体工作负载、异构设备以及不同网络和执行条件下的泛化能力。

参考文献

[1] A. Borzunov、D. Baranchuk、T. Dettmers、M. Riabinin、Y. Belkada、A. Chumachenko、P. Samygin与C. Raffel，“Petals：大模型的协作推理与微调”，载于第*61*届计算语言学协会年会论文集（第*3*卷：系统演示），第558–568页，计算语言学协会，2023年。[2] M. Zhang、X. Shen、J. Cao、Z. Cui与S. Jiang，“EdgeShard：通过协同边缘计算实现高效大语言模型推理”，*IEEE*物联网期刊，第12卷，第10期，第13119–13131页，2025年。[3] S. Kim、S. Moon、R. Tabrizi、N. Lee、M. W. Mahoney、K. Keutzer与A. Gholami，“面向并行函数调用的大语言模型编译器”，载于第*41*届国际机器学习会议论文集，机器学习研究会议录第235卷，第24370–24391页，PMLR，2024年。[4] J. Zhang、J. Xiang、Z. Yu、F. Teng、X.-H. Chen、J. Chen、M. Zhuge、X. Cheng、S. Hong、J. Wang、B. Zheng、B. Liu、Y. Luo与C. Wu，“AFlow：智能体工作流生成的自动化”，载于第十三届国际学习表征会议，第34040–34077页，2025年。[5] F. Strati、S. McAllister、A. Phanishayee、J. Tarnawski与A. Klimovic，“D´ej`avu：面向快速容错生成式大语言模型服务的KV 缓存流式传输”，载于第*41*届国际机器学习会议论文集，机器学习研究会议录第235卷，第46745–46771页，PMLR，2024年。[6] M. Zhang、J. Cao、T. Wu、X. Chen与N. Wang，“协同边缘计算中地理分布式大语言模型推理的高效KV缓存迁移”，载于*2026*年*IEEE*第*46*届分布式计算系统国际会议（*ICDCS*），第272–282页，2026年6月。ISSN：2575-8411。[7] Z. Cao、S. Han、J. Zhang、Y. Ren、Y. Li与P. P. C. Lee，“LUMEN：分布式大语言模型服务的协调式故障恢复”，2026年。[8] R. Devaraj、P. P. Nair与S. Moulik，“DARCS：异构边缘基础设施上基于大语言模型的智能体工作流的截止期限与可靠性约束调度”，系统体系结构期刊，第181卷，第104019页，2026年12月。[9] D. Hafner、T. Lillicrap、I. Fischer、R. Villegas、D. Ha、H. Lee与J. Davidson，“学习潜在动力学以进行基于像素的规划”，载于第*36*届国际机器学习会议论文集，机器学习研究会议录第97卷，第2555–2565页，PMLR，2019年。[10] S. Yao、J. Zhao、D. Yu、N. Du、I. Shafran、K. Narasimhan与Y. Cao，“ReAct：语言模型中推理与行动的协同”，载于第十一届国际学习表征会议，2023年。[11] T. Schick、J. Dwivedi-Yu、R. Dessi、R. Raileanu、M. Lomeli、E. Hambro、L. Zettlemoyer、N. Cancedda与T. Scialom，“Toolformer：语言模型可以自学使用工具”，载于神经信息处理系统进展，第36卷，68551，2023年，第68539页–。[12] N. Shinn、F. Cassano、A. Gopinath、K. Narasimhan与S. Yao，“Reflexion：具备言语强化学习的语言智能体”，载于神经信息处理系统进展，第36卷，第8634–8652页，2023年。[13] Q. Wu、G. Bansal、J. Zhang、Y. Wu、B. Li、E. Zhu、L. Jiang、X. Zhang、S. Zhang、J. Liu、A. Awadallah、R. W. White、D. Burger与C. Wang，“AutoGen：通过多智能体对话赋能下一代大语言模型应用”，载于第一届语言建模会议，2024年。[14] S. Hong、M. Zhuge、J. Chen、X. Zheng、Y. Cheng、J. Wang、C. Zhang、Z. Wang、S. Yau、Z. Lin、L. Zhou、C. Ran、L. Xiao、C. Wu与J. Schmidhuber，“MetaGPT：面向多智能体协作框架的元编程”，载于第十二届国际学习表征会议，第23247–23275页，2024年。

[15] A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y.-X. Wang，《语言智能体树搜索统一了语言模型中的推理、行动与规划》，载于第*41*届国际机器学习会议论文集，第235卷机器学习研究会议录，第62138–62160页，PMLR，2024年。[16] S. Hu, C. Lu, and J. Clune，《智能体系统自动化设计》，载于第十三届国际学习表征会议，第21344–21377页，2025年。[17] G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. Yu, and T. Chen，《少说废话：面向基于大语言模型的多智能体系统的经济型通信流水线》，载于第十三届国际学习表征会议，第75389–75428页，2025年。[18] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica，《Ray：面向新兴人工智能应用的分布式框架》，载于第*13*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 18*），第561–577页，USENIX 协会，2018年。[19] H. Zhang, A. Cardoza, P. B. Chen, S. Angel, and V. Liu，《容错事务性有状态无服务器工作流》，载于第*14*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 20*），第1187–1204页，USENIX 协会，2020年。[20] Z. Jia a nd E. Witchel，《Boki：基于共享日志的有状态无服务器计算》，载于第*28*届 *ACM* 操作系统原理研讨会论文集，第691–707页，计算机协会，2021年。[21] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica，《利用 PagedAttention 高效管理大语言模型服务的内存》，载于第*29*届操作系统原理研讨会论文集，第611–626页，计算机协会，2023年。[22] B. Sun, Z. Huang, H. Zhao, W. Xiao, X. Zhang, Y. Li, and W. Lin，《Llumnix：大语言模型服务的动态调度》，载于第*18*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第173–191页，USENIX 协会，2024年。[23] Y. Fu, L. Xue, Y. Huang, A.-O. Brabete, D. Ustiugov, Y. Patel, and L. Mai，《ServerlessLLM：大语言模型的低延迟无服务器推理》，载于第*18*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第135–153页，USENIX 协会，2024年。[24] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, and Z. Jia，《Spot-Serve：在可抢占实例上服务生成式大语言模型》，载于第*29*届 *ACM* 编程语言与操作系统架构支持国际会议论文集，第*2*卷，第1112–1127页，计算机协会，2024年。[25] S. Qin, D. Pi, Z. Shao, Y. Xu, and Y. Chen，《面向多云系统工作流调度问题的可靠性感知多目标模因算法》，*IEEE*并行与分布式系统汇刊，第34卷，第4期，第1343–1361页，2023年。[26] A. Taghinezhad-Niar and J. Taheri，《边缘云连续体中面向能量与截止期限约束的物联网工作流的容错、成本高效调度》，*IEEE* 服务计算汇刊，第18卷，第5期，第2892–2903页，2025年。[27] Z. Li, L. Zheng, Y. Zhong, V. Liu, Y. Sheng, X. Jin, Y. Huang, Z. Chen, H. Zhang, J. E. Gonzalez, and I. Stoica，《AlpaServe：结合模型并行的统计复用，用于深度学习服务》，载于第*17*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 23*），第663–679页，USENIX 协会，2023年。[28] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang，《DistServe：分离预填充与解码，以实现有效吞吐量优化的大语言模型服务》，载于第*18*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第193–210页，USENIX 协会，2024年。[29] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee，《利用 Sarathi-Serve 调和大语言模型推理中的吞吐量与延迟权衡》，载于第*18*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第117–134页，USENIX 协会，2024年。[30] C. Lin, Z. Han, C. Zhang, Y. Yang, F. Yang, C. Chen, and L. Qiu，《Parrot：利用语义变量高效服务基于大语言模型的应用》，载于第*18*届 *USENIX* 操作系统设计与实现研讨会（*OSDI 24*），第929–945页，USENIX 协会，2024年。[31] A. Tumanov, T. Zhu, J. W. Park, M. A. Kozuch, M. Harchol-Balter, and G. R. Ganger，《TetriSched：采用自适应

plan-ahead in d动态 h异构 c]群集，”载于会议录 *d*录 *o f* 第十一届欧洲计算机系统会议，第 1–16 页，计算机协会，2016 年。[32] Y. Zhang、W. Hua、Z. Zhou、G. E. Suh 和 C. Delimitrou，“Sinan：基于机器学习且具备 QoS 感知能力的云微服务资源管理”，载于第 *26* 届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集，第 167–181 页，计算机协会，2021 年。[33] Z. Zhou、Y. Zhang 和 C. Delimitrou，“AQUATOPE：面向多阶段无服务器工作流的 QoS 与不确定性感知资源管理”，载于第 *28* 届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集，第 *1* 卷，第 1–14 页，计算机协会，2023 年。[34] C. C. White，“有限时域、部分可观测半马尔可夫优化问题的求解程序”，运筹学，第 24 卷，第 2 期，第 348–358 页，1976 年。[35] M. Zhang 和 M. Revie，“用于机器维护的连续观察部分可观测半马尔可夫决策过程”，*IEEE* 可靠性汇刊，第 66 卷，218，2017 1，202期，第 – 页。[36] C. G. Cassandras 和 S. Lafortune，离散事件系统导论。Springer，第 2 版，2008 年。[37] K. Chua、R. Calandra、R. McAllister 和 S. Levine，“利用概率动力学模型进行少量试验的深度强化学习”，载于神经信息处理系统进展，第 31 卷，Curran Associates, Inc.，2018 年。[38] C. E. Jimenez、J. Yang、A. Wettig、S. Yao、K. Pei、O. Press 和 K. R. Narasimhan，“SWE-bench：语言模型能否解决现实世界中的 GitHub 问题？”，载于第十二届国际学习表征会议，2024 年。[39] H. Liu、J. Cao、B. Yang、D. Bai、Y. Cao、X. Shen、Y. Zhang、J. Liang、S. Jiang 和 M. Zhang，“PolyLink：一种基于区块链的去中心化边缘 AI 平台，用于大语言模型推理”，载于*2025* 年 *IEEE* 区块链国际会议（*Blockchain*），第 101–108 页，IEEE，2025 年。

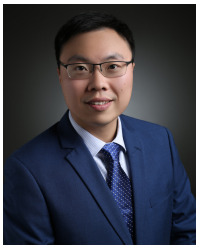


Xiangchun Chen现为中国香港特别行政区香港理工大学计算学系博士后研究员。他于2021年获得中国哈尔滨工业大学计算机科学与技术专业工学学士学位，并于2026年获得中国香港特别行政区香港理工大学计算学博士学位。他的研究兴趣包括边缘人工智能代理、边缘大语言模型、强化学习和数字孪生。



Jiannong Cao (Fellow, IEEE) 现任香港理工大学潘宗光慈善基金会数据科学讲座教授，以及计算学系分布式与移动计算讲座教授。他是香港理工大学物联网人工智能研究院的创始院长，也是互联网与移动计算实验室主任。他还担任香港理工大学副校长（教育）。他曾于……起担任该学系主任

2011年至2017年。曹教授是欧洲科学院成员、IEEE会士、中国计算机学会（CCF）会士及ACM杰出会员。他的研究兴趣包括分布式系统与区块链、无线感知与网络技术、大数据与机器学习，以及移动云和边缘计算。



Longxiang Gao 于2014年获得澳大利亚迪肯大学计算机科学博士学位。他目前是中国济南齐鲁工业大学（山东省科学院）山东省计算机科学中心教授。他曾任澳大利亚IBM研究与开发部博士后研究员，以及迪肯大学讲师和高级讲师。他的研究兴趣包括边缘智能、网络安全和量子计算。他是一名高级

ACM 和 IEEE 的成员。



Xiaoming Wu 于 2006 年获得中国山东大学计算机科学与技术工程硕士学位，并于 2017 年获得山东科技大学软件工程博士学位。他目前是齐鲁工业大学（山东省科学院）山东省计算机科学中心的教授，工作地点位于中国济南。他还担任计算机科学与技术学院和山东省计算机科学中心主任。

他的研究兴趣包括网络空间安全、工业互联网、数据安全和隐私保护。



Yusheng Ji（IEEE 会士）于东京大学获得电气工程学士、硕士和工学博士学位。她于 1990 年加入日本东京的国家科学信息系统中心（NACSIS）。她目前是日本东京国立信息学研究所（NII）的名誉教授。她的研究兴趣包括网络资源管理和移动计算。