中国科学 : 信息科学 2025 年 第 55 卷 第 3 期: 464–480
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
先进计算与新兴软件专题 · 论文

人机物融合泛在应用的系统支撑

曹云帆[1,2], 赵超懿[1,2], 刘瀚之[1,2], 王加益[1,2], 王慧妍[1,2], 余萍[1,2], 曹春[1,2], 许畅[1,2], 马晓星[1,2], 蒋炎岩[1,2*]

1. 计算机软件新技术全国重点实验室, 南京 210023
2. 南京大学计算机学院, 南京 210023
* 通信作者. E-mail: jyy@nju.edu.cn

收稿日期: 2024–11–10; 修回日期: 2025–01–14; 接受日期: 2025–02–18; 网络出版日期: 2025–02–27

科技部重点研发项目 (批准号: 2022YFB4501800) 资助

摘要 泛在计算为人机物融合应用带来了新的可能性, 但也面临着异构资源管理、需求多样性和计算资源受限等问题; 同时, 人工智能模型的快速发展也为智能应用的开发带来了机遇与挑战. 为此, 本文提出了元级化软件定义的泛在操作系统设计理念, 通过引入 “软件孪生” 技术实现对异构资源的抽象管理, 采用 “上传下达” 的层级结构应对复杂需求的分解, 并通过智能原生设计实现对人工智能模型的支持. 本文进一步以操作系统智能助手、“天网” 监控系统和疾病防控系统 3 个典型泛在计算应用场景为例, 阐述了本文所提出的泛在操作系统设计理念在实践中的优势和潜在价值, 预示着智能时代下人机物融合的广泛应用前景.

关键词 泛在计算, 泛在操作系统, 软件工程

1 引言

随着物联网 (Internet of Things, IoT) 设备的数量迅速增长和数据交互需求的不断增加, 计算能力已广泛嵌入到各种设备和环境中, 实现无缝的计算服务, 支持人、计算设备与物理世界的深度融合的泛在计算 (ubiquitous computing) [1] 逐渐成为新时代应用开发和技术发展的重要驱动力之一. 当前人工智能的高速发展为人机物融合泛在应用带来了新的机遇, 但从人工智能模型到智能化人机物融合应用依然存在距离, 人工智能模型仍面临难控制、难使用、难解释的困境, 人机物融合泛在应用仍普遍存在编程、维护、移植瓶颈, 从而限制了相关产业的智能化、数字化转型.

针对人机物融合应用特有的开放异构资源和多样化的需求, 现有的操作系统通常采用的有明确规约、由设计者决定的实体抽象 (比如文件、进程), 难以对复杂异构设备进行有效统一的抽象和简化; 而现有的中间件和应用框架虽然在特定领域有所成效, 但往往仅集中于特定场景的具体功能设计, 难以为复杂多样的人机物融合泛在应用提供元级解决方案. 因此, 为解决人机物融合应用开发与运行支持,

引用格式: 曹云帆, 赵超懿, 刘瀚之, 等. 人机物融合泛在应用的系统支撑. 中国科学: 信息科学, 2025, 55: 464–480, doi: 10.1360/SSI-2024-0338
Cao Y F, Zhao C Y, Liu H Z, et al. System support for ubiquitous human-cyber-physical fusion applications. Sci Sin Inform, 2025, 55: 464–480, doi: 10.1360/SSI-2024-0338

基于操作系统 “对下管理资源, 对上提供服务” [2] 的基本属性, 应针对泛在场景开发新形态的操作系统, 即泛在操作系统 [3]. 为此, 本文提出了元级化软件定义的泛在操作系统, 旨在解决泛在计算场景下的资源分配、数据管理、智能赋能等问题, 从基础上为人机物融合泛在应用提供系统支撑.

本文提出的元级化软件定义的泛在操作系统以软件孪生为基础能力, 数据上传和指令下达为基本形态, 智能原生为基本设计. 元级化软件定义的泛在操作系统: (1) 通过软件定义实现对于人机物融合泛在应用异构资源的统一抽象, 为应用各类实体建立软件孪生并在此基础上构建全系统状态, 从而支持软件开发过程从 “面向需求完成设计实现” 转换至 “在应用场景全量日志数据上做出决策” 的新范式; (2) 参考 UNIX 管道的基本设计思想, 为泛在场景设计了新形态的沟通管道, 并以此为途径将人机物融合应用自然划分为若干可复用可交互可协作的层级组件 (智能体), 通过树状组件结构, 支持上层节点与下层节点之间指令下达及数据上传; (3) 规范化定义应用组件间的数据和命令接口, 规范包含时间 (when)、地点 (where)、人物 (whom)、事件 (what) 的单点数据抽象并明确组件处理数据的输入输出规约, 支持基于新时代智能技术完成组件功能实现, 实现低代码编程.

本文组织如下: 首先从人工智能时代泛在计算的现状和前景出发分析人机物融合应用的机遇和挑战, 并从操作系统的发展历史中凝练人工智能时代泛在操作系统的新需求, 进而提出元级化软件定义的泛在操作系统的基础能力、基本形态和基准设计. 最后, 本文以操作系统智能助手、“天网” 监控工程和疾病防控 3 个典型场景为例, 讨论了泛在场景开发人机物融合应用的挑战及困难, 并介绍了元级化软件定义的泛在操作系统如何支撑上述典型场景的应用开发. 本文旨在为泛在计算背景下操作系统发展提供新的视角, 并揭示新智能时代泛在操作系统的潜在价值和广泛应用前景.

2　人工智能时代人机物融合泛在计算的新蓝海

2.1　泛在计算: 现状、前景与困难

随着低成本、微型化计算机硬件的快速发展, 互联网已经深入人类社会和物理世界的各个方面. IDC 发布的《中国物联网连接规模预测, 2023–2027》[1)] 预计 2023 年中国物联网连接量超 66 亿, 未来 5 年复合增长率约 16.4%, 一个万物互联的人机物融合泛在计算时代正在开启 [4]. 所谓泛在计算 [1], 指计算广泛分布于物理环境中, 无处不在、无迹可寻.

数量庞大、深度嵌入物理环境的计算设备为人机物融合泛在应用带来了广阔的前景. 基于场景内广泛分布的感知、计算和通信能力, 应用可以数字化地管理和调度软硬件资源, 最终实现人类、计算设备和物理设备的无缝融合, 满足生产生活需求. 人机物融合泛在计算模式改变了既有行业形态, 催生了新兴业务的出现, 其中软件成为应用价值观的主要载体. 以工业物联网为例 [5], 一方面, 制造业巨头们纷纷推出自己的工业物联网平台, 包括西门子公司的 MindSphere、通用电气 (GE) 公司的 Predix、航天科工集团的航天云网等, 力图打造为自身服务的软件; 另一方面, 国内外信息产业巨头借助自身在软件领域优势推出通用工业物联网平台, 如 Microsoft Azure IoT Platform、Amazon AWS IoT Platform、阿里云 IoT、华为 Fusion Plant 等.

与此同时, 人机物融合应用所面对的开放非确定性环境带来了新一轮的 “软件危机”, 软硬件资源的有效调度管理和软件高效开发、可信保障面临挑战, 主要难点体现在以下方面.

• 泛在环境的复杂性. 泛在场景中, 数量庞大的计算设备与物理设备紧密相连, 具有专用性和特异性, 加之 “人在回路” 带来的复杂影响, 环境多元复杂且高度不可预测.

• 业务需求的多样性. 泛在场景涵盖广泛, 需求复杂多变, 不会有 “大一统” 的通用、普适的解决方案 [6], 系统需灵活适应不同领域的业务需求.

1) https://www.idc.com/getdoc.jsp?containerId=CHC50360723

• 高度异构的计算资源. 泛在场景需打破从极低能耗的传感终端到边、云、数据中心高度异构资源之间的信息壁垒, 从而实现资源的智能高效协同和调度.

• 可信与安全保障. 万物互联导致系统前所未有的开放, 需要内构安全机制以识别和阻断异常行为, 维护数据安全和隐私 [6].

2.2　人工智能时代下人机物融合应用的机遇与挑战

近年来, 以深度学习、大语言模型为代表的人工智能技术飞速发展, 人工智能模型的性能持续增强, 促进新一代智能应用井喷式增长, 涉及工业 [7]、农业 [8]、医疗 [9]、商务 [10] 等多个领域. 机器学习模型是这些智能应用的核心, 基本原理是通过对大规模训练数据集进行统计学习, 构建从输入空间到输出空间的映射函数 [11~13]. 在训练阶段, 模型利用随机梯度下降等优化算法 [14], 通过迭代优化参数以最小化损失函数, 从而捕获数据中蕴含的统计规律与潜在模式. 在推理阶段, 经过训练的模型能够对符合训练数据分布的输入数据进行处理, 输出相应的预测结果.

人工智能模型的引入极大地增强了人机物融合系统的泛用性和有效性. 一方面, 数据驱动的学习范式赋予人工智能系统从海量训练样本中提取统计规律和潜在模式的能力, 使其在训练数据分布范围内具备一定的泛化性能, 即能够对未曾观测的同分布数据进行有效处理. 这种自适应机制使人工智能系统能够自动学习复杂的特征表示和决策规则, 从而在一定程度上实现对多源异构数据的有效处理和开放环境的适应. 另一方面, 人工智能模型的概率性本质使得模型能够自然地表达和处理物理世界中普遍存在的不确定性, 为人机物融合应用在开放、动态、非确定性环境下的智能决策提供了重要的解决路径.

然而, 人工智能模型的引入也带来了新的挑战. 首先, 尽管人工智能模型在一定范围内表现出了优异的泛化性能, 但这种能力并不是无限的. 训练数据的分布决定了人工智能模型能力的边界, 在面对分布外 (out-of-distribution, OOD) 场景时, 模型的性能会出现严重退化 [15, 16]. 针对这一固有的局限性, 一种基础性的技术方案是将模型的推理过程严格限定在其训练分布范围内. 这要求对输入数据进行精确分类, 并为每类数据分布匹配相应的专用模型. 因此, 为了在人机物融合场景中充分利用人工智能模型, 需要系统性地将复杂的应用需求分解为一系列规模可控且符合模型训练分布的子任务, 以最大程度地保证模型在同分布 (in-distribution, ID) 数据上进行推理, 从而基于 “分而治之” 的方法构建智能应用的整体解决方案.

其次, 人工智能模型输出结果的概率性特征与编程语言严格的形式语义之间存在本质差异, 这为开发者在融合两种范式构建智能应用时带来了严峻的挑战. 编程语言的语义具有确定性且由开发者完全掌控; 与之相对, 人工智能模型的概率性本质导致其泛化性难以度量, 输入的微小扰动可能产生大相径庭的输出. 目前仅提供训练、推理 API 的机器学习框架难以面对智能化人机物融合应用开发的挑战: 带有人工智能模型软件的调试、测试和性能评估本身就是软件工程领域公认的难题, 加之应用为提升系统可靠性普遍包含复杂的后处理机制在运行时持续监测异常, 智能化人机物融合应用的开发亟需系统化的解决方案高效管理深度耦合的代码与人工智能模型.

学件 (learnware) [17] 是支撑人机物融合泛在智能化应用的重要尝试, 其通过规约 (specification) 描述模型并屏蔽其底层细节, 将模型集成在学件坞中, 并根据不同学件擅长的领域分解输入域, 使每个具体的输入都能由最合适的模型进行处理, 最终为用户提供合适的模型. 学件提供了智能应用的新编程范式, 实现了调用人工智能的机制 (规约) 和策略 (模型选取和组合) 分离的重大突破, 并且有潜力进一步集成大语言模型以及云端推理等功能. 然而, 学件解决的问题范围仍局限于软件代码单次调用人工智能 – 返回结果的局部流程. 智能应用开发者通常可以轻易说明高层次的业务需求和设计, 但难以直接描述每个模型的具体需求, 因而学件并不能解决所有问题. 人工智能如何服务人机物融合场景的环境复杂性、业务逻辑多样性、应用需求的模糊性仍面临挑战.

综上所述, 人工智能技术对数据复杂特征的自适应能力和对开放非确定环境的处理能力为人机物

融合泛在应用在数据分析与智能决策等领域提供了重要机遇. 然而, 智能模型在面对分布外场景时性能显著下降的固有局限性需要开发者对应用的复杂需求进行分解, 且模型输出结果的概率性本质与编程语言精确的形式语义间存在巨大鸿沟, 这些都为智能化应用的开发带来了巨大挑战. 为有效降低智能化人机物融合应用的开发复杂度, 需进一步实现人工智能模型的执行机制与调度策略的解耦, 为智能应用开发提供更高层次的系统化支持.

3　从操作系统到泛在操作系统: 新思考

3.1　操作系统发展回顾

操作系统是计算系统的核心, 向下管理硬件资源, 向上为用户和应用程序提供服务支撑 [2]. 操作系统的出现使开发者能摆脱复杂的底层硬件机制、专注于应用设计, 促进了应用生态的繁荣, 并最终推动了计算机从专用走向普及. 操作系统发展的重大变迁存在 “20 年周期律” [6], 即每 20 年左右会出现一次突破性进展, 诞生新一代的操作系统, 且装机数量和用户规模上相比于旧生态扩展 1∼2 个数量级. 具体而言, 操作系统的发展可分为以 IBM OS/360 和 UNIX 等操作系统为代表的主机时代, 以 Windows, macOS 和 Linux 为代表的个人计算时代, 以及以 Android 和 iOS 为代表的移动计算时代. 操作系统发展的 “20 年周期律” 来源于计算机产业发展的 “贝尔定律” [18], 即计算设备每 10 年完成一次升级换代, 数量至少增加一个数量级; 新型计算模式与计算设备共同催生了新型应用, 这些应用需要新型操作系统实现底层复杂性的屏蔽.

操作系统领域的重要里程碑是 Ken Thompson 和 Dennis M. Ritchie 设计与实现的 UNIX 操作系统, 二人因此获得了 1983 年的图灵奖. UNIX 保留了 MULTICS 中进程、文件系统、流式输入输出、设备文件等重要抽象, 并实现了管道机制实现应用程序的互联, 这些特性对计算机软件产业产生了深远的影响. 从软件工程的角度, 管道实现了应用程序之间的协同, 它鼓励用户将复杂的需求逐级分解, 直至单个命令行程序能够处理; 以文本形式在管道间流转的数据在开发者社区形成了约定俗成的数据接口, 实现了应用间的互联互通; 管道同时也肩负了组件之间的同步和流量控制功能. UNIX 生态中的命令行工具的设计理念始终是 “能被其他应用所用”, 赋予了组合工具的扩展性, 促进了 UNIX 生态的繁荣.

UNIX 催生了操作系统标准 POSIX, 实现了世界范围统一的系统调用、库函数和进程间通信标准, 支撑海量应用程序的无缝移植. POSIX 奠定了操作系统核心基础层抽象, 现有所有主流操作系统都基于 POSIX 开发, Linux, Windows 和 macOS (和基于这些操作系统的应用系统, 例如 Android 和 iOS) 均为应用程序提供了 POSIX 核心基础 API. Fuchsia OS [19] 等新兴操作系统也通过兼容库等方式实现了对 POSIX 的支持.

计算需求的增加和网络技术的发展促进了分布式系统的诞生和演化 [20]. 通过将计算任务分散到多个独立的计算机节点上, 分布式系统能够突破单机计算的能力限制, 带来大规模并行和资源共享的机遇, 然而计算资源和集群网络的不可靠也为分布式系统的开发和维护带来了严峻的挑战. 为此, GFS [21], ZooKeeper [22], Plan 9 [23] 等分布式系统在共识协议的基础上为计算资源构建抽象, 在不可靠网络和计算机硬件的基础上为上层应用提供近似单机环境的高可用 API.

在此浪潮下, 虚拟化技术的进步 [24] 与大规模数据中心的诞生 [25] 开启了云计算时代, 为移动操作系统的普及奠定了基础 [26]: 云服务提供商统一管理大规模的设备集群, 通过虚拟化技术将物理设备抽象为虚拟的计算资源, 向用户提供可按需付费和弹性扩展, 屏蔽了底层系统的维护细节 [27,28]. 事实上, 尽管名义上与操作系统无关, 现代的数据中心已经具有与操作系统高度相似的职能 [25], 向下管理异构的软硬件资源 (服务器、CPU、硬盘等), 向上为高层应用提供服务 (算力、存储、网络等资源可配置的虚拟计算机).

从现代操作系统的发展历程可以看到, 技术的发展引起的新兴计算需求会带来新的机遇; 伴随着计算资源的进一步异化和复杂化, 操作系统的概念也随之扩展和泛化. 因此, 泛在计算时代的开启势必带来对新形态操作系统的需求, 以应对人机物融合应用生态发展的诸多挑战.

3.2　人工智能时代泛在计算场景下的新型操作系统

在面向人机物融合泛在计算的新兴计算模式中, 应用场景内高度互联的设备和实体组成了有机的整体, 应用场景本身即可被视为一种新形态的计算机, 即场景计算机 [29]; 场景计算机对软件提出了可定制、柔性化、可演化、自适应、智能化等全新需求. 场景计算机是新兴计算设备和计算模式发展的动力来源, 但相应的软件开发革命仍未完成. 基于操作系统的 “20 年周期律”, 新形态操作系统将面向泛在场景内的新型计算设备, 支持新型的泛在计算模式与智能计算模式, 并成为泛在场景下人机物融合应用开发瓶颈的解决方案. 沿袭以计算模式分类操作系统的惯例, 此类新型操作系统可称为 “泛在操作系统” [3].

相比于传统操作系统、云计算系统等系统, 场景计算机管理的软硬件资源呈现显著的异构特征, 来自不同厂商、不同型号的异构设备会使用不同的传输和控制协议, 有些设备几乎不具备开放的扩展接口. 同时, 由于泛在场景前所未有的动态特性, 系统内实体通常不完全由设计者决定, 而是随环境动态变化, 可能在任何时间出现或消失, 并且有可能尝试执行超出预期甚至非法的操作, 呈现出高度不可预测的特征. 高度开放的场景还意味着资源总量难以预测和控制, 可能出现设备集中聚集等资源调度挑战. 因此, 泛在操作系统需适应开放的环境, 管理海量与受限的资源, 屏蔽软硬件资源的异构性, 向上层应用提供易于编程的接口.

人机物融合泛在应用具有其鲜明特征: 泛在环境异构时间序列数据源天然存在噪声, 根据不同数据来源作出的决策可能互相冲突. 开发者通常无法预知开放环境中各个领域、不同类型的所有需求, 即便需求明确, 有时也难以通过预设逻辑进行处理.

因此, 如直接使用底层操作系统提供的原子 API (如 POSIX API) 进行人机物融合应用开发, 难免出现大量代码冗余, 带来软件维护危机. 工业界普遍采用了 “自底向上逐层抽象” 的系统构建范式, 通过中间件和应用框架简化软件开发. 例如, InfluxDB [30], Prometheus [31], IoTDB [32] 等数据库实现了高效的时序数据管理, Apache Hadoop [33] 等分布式系统实现了多节点协作, AWS Lambda [27] 等 serverless 框架屏蔽了云计算的服务器管理需求. 然而, 这些中间件和框架通常局限于特定功能, 缺乏全局视角, 难以利用全局信息. 新兴的泛在操作系统可从更高视角审视应用需求, 吸收现有实践的成功经验并突破面向单一功能的局限, 为人机物融合泛在计算应用提供多种功能的支持.

此外, 随着智能时代的到来, 智能应用的需求大幅增加, 在人机物融合的泛在计算场景下, 智能化应用的开发、运行和管理面临新的挑战. 受限于历史等原因, 现有的成熟操作系统缺乏原生的智能能力支持, 难以充分满足智能应用的需求. 为支持智能应用的开发, 泛在操作系统需实现智能原生: 支持智能应用的问题规模分解和多模型集成, 满足模糊需求, 在全局视角统一管理智能模型的运行, 并容忍模型产生的误差数据.

综上所述, 泛在操作系统应具有如下能力.

(1) 异构资源抽象. 泛在操作系统需实现异构资源的抽象, 解决专用性和特异性问题, 实现通用且简洁的资源控制.

(2) 应用需求分解. 泛在操作系统需提供合适的应用开发范式, 帮助开发者分解多样的业务需求, 并将问题规模拆解至人工智能模型可解决的范围.

(3) 智能原生支持. 泛在操作系统应当从设计和构建阶段就具备智能化能力的系统和应用, 并实现对模型误差数据的自适应以及模糊需求的支持.

理想的泛在操作系统应当能解决智能化人机物融合应用开发面临的困难, 屏蔽底层复杂性, 并为应用提供充分的支持. 满足上述能力的泛在操作系统可以解决智能化人机物融合应用开发的关键瓶

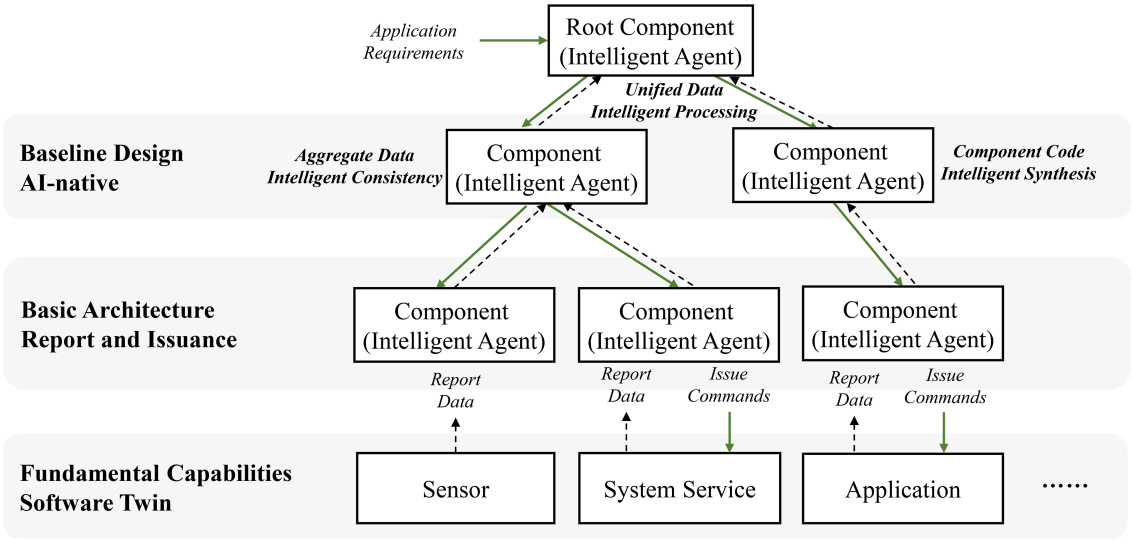

图 1　(网络版彩图) 元级化软件定义的泛在操作系统架构.

Figure 1　(Color online) Meta-level software-defined ubiquitous operating system architecture.

颈, 从而显著减轻人机物融合泛在应用的开发困难, 带来应用生态的繁荣.

4　元级化软件定义的泛在操作系统

“操作系统” 的主要职责是 “管理硬件资源、为上层应用提供服务”. 在泛在计算场景下, 被管理的资源不再是操作系统设计者决定、具有明确规约的对象 (如进程、文件等), 而是开放环境下的异构资源系统, 而传统操作系统未能对这些复杂资源作出有效抽象与简化. 为应对 “应用场景就是计算机” 所带来的应用开发挑战, 本文分析了泛在计算场景中的应用开发新范式, 借鉴人类社会在演进中自然形成的管理架构, 提出支撑泛在计算的元级化软件定义泛在操作系统.

元级化软件定义的泛在操作系统的核心理念是将软件系统抽象为具备系统内外状态感知和调控能力的泛在智能体 (agent) [34], 泛在操作系统作为其运行支撑, 不仅承担其生命周期管理, 亦负责其间的数据通信和规范化处理, 实现智能体之间的高效、可靠协同, 并同时无缝兼容传统软件工程组件 (代码模块) 和 “具有信念、期望、意图和自主行为能力” 的传统智能体 [35~37], 如基于大语言模型实现的推理 – 计划 – 行动智能体 [38]. 本文认为, 人机物融合应用构建的元级化方法包含三大主要理念: 在上帝视角观测应用场景的基础能力、拟人类社会组织方式的应用基本形态, 以及面向人工智能模型的基准设计, 如图 1 所示. 这些理念阐述如下.

4.1　基础能力: 软件孪生

自软件工程学科诞生以来, 即便进入人工智能时代, 无论采取何种开发范式, 软件系统实现遵循的一般规律仍是 “开发团队负责将需求、设计文档、报告等转换为可运行、高质量的代码实现”. 在此过程中, POSIX, Android Platform, Apache Flink [39] 等操作系统和应用中间件应运而生, 使用 “对象” 统一建模软件世界中的实体 (如 POSIX 中的进程、文件; Android 中的 Activity, Broadcast Receiver; Flink 中的 DataStream, Transformation 等) 并用 API 建模实体上的操作, 屏蔽被管理对象的复杂细节. 然而, 这些 API 的主要设计范式是 “对单个确定的对象实施单个确定的操作”, 在人机物融合场景中, 异构资源使面向 API 的编程繁琐、低效、易错, 即泛在操作系统需要对计算场景进行高层抽象, 提供易于管理异构资源的编程模型.

为破解人机物融合异构资源难编程管理之挑战, 本文提出泛在操作系统的基础能力是构建随时间演化的全系统状态 “上帝视角”, 以便应用程序随时按需获取其所需信息, 并组织成高效的信息处理网格, 及时对信息进行加工、处理并将与需求无关的数据丢弃, 以应对海量、冗余、不可靠的运行时数据.

在概念和方法学层面, 本文倡议软件开发过程从 “面向需求完成设计实现” 到 “在应用场景全量日志数据上做出决策” 的思维方式转换, 即 “软件孪生” 的软件方法学. 这一思路来自于以下观察: 人机物融合应用场景中的一切人和物理实体的状态终究是通过软件 (计算机) 感知的. 假想有一台处理速度无限、存储容量无限、传输带宽无限的 “理想计算机”, 获得了系统中一切软件执行的细粒度信息 (不妨假设细到每条处理器指令执行的时间戳、操作数, 以及这些指令到源代码实现和需求的对应), 则这一计算机已经具备处理人机物融合应用需求的全部信息, 只需做出正确决策即可.

在系统支撑层面, 泛在操作系统的主要任务是实现 “面向全量数据做减法” 的编程机制, 使开发者摆脱 “面向需求分析做加法” 的桎梏, 将设计目标转换到在全量数据中筛选出与应用需求相关的部分, 并在此基础上设计决策逻辑的新编程范式. 面向实际的应用需求, 通常只需在理想计算机获取的小部分数据中提取信息即可做出正确决策. 因此这一理念天然将物理空间实体 “可感知、可控制、有冗余、不可靠” 的特性纳入软件设计阶段的考量, 从而实现对不稳定、不确定环境的原生支持. 泛在操作系统则可对软件孪生世界映像的构建, 包括数据的获取、控制的下发、组件的部署等提供系统化的支撑, 将传统的软件部件转换为具备一定能动性、可调配的泛在智能体. 近二十年来系统软件技术的发展为软件孪生的实现奠定了坚实的基础, 使我们拥有技术手段在已有的操作系统、中间件和软件上构建提取其运行时状态的通用 “平行层”. 从 PIN 为代表的二进制插桩, 到实现了 Linux 内核的用户态可观测和可编程的 eBPF, 以及主流编程语言虚拟机实现的工具接口 (如 JVMTI), 均可用于实现应用无感知的插桩机制, 即将已有成熟技术用于实现软件的 “平行孪生层”, 并通过提取需求相关的软件数据进行泛在应用的决策.

4.2　基本形态: 上传下达

软件孪生打破了软件之间的信息孤岛, 提供了 “系统之系统” 的全局视角. 与传统软件 (直接执行程序代码、调用 API 修改程序状态完成需求) 不同, 人机物融合应用中涌现的需求通常需要系统中的资源响应当前观测的系统状态, 并在系统的各个部分分别做出适当的决策, 将整个系统的状态逐渐迁移到满足需求的状态. 传统软件工程方法使用紧密耦合代码集中控制资源, 而随着场景计算机中的资源走向动态异构, 紧密耦合的代码必将成为软件开发、维护、演化中的掣肘. 将应用需求分解到规模较小、逻辑简单、松散耦合、协同工作的逻辑组件是实现人机物融合应用的必要步骤.

为破解人机物融合应用复杂性分解之挑战, 本文提出泛在操作系统应为应用程序作出基本形态的约束, 形成类似多智能体上传下达高效协同工作的类社会组织形态, 同时实现应用需求的高效分解和软件复杂性的控制. 泛在操作系统完成对智能体的运行时管理: 系统中的每一基本组件 (泛在智能体) 都既作为接收需求的 “执行者”, 也作为分解需求的 “领导者”, 将应用的顶层需求逐层传递给底层效应器.

本文注意到操作系统领域对组件协同需求不断增长, 并终将面临与人类社会发展类似的瓶颈. 与人类社会自上而下的政府组织、协同合作的国家机器类似, 随着人机物融合复杂性的增长, 逐层分解、自我管辖、共同工作的组织形态是管理复杂系统的有效途径. 早期的批处理系统仅通过文件系统交换数据. UNIX 管道极大降低了进程间的协同的门槛, 引领了一场操作系统革命. 在面向更复杂、开放的移动计算场景中, Android 引入了 Binder 底层机制, 并进一步提供了 Intent, Content Provider 实现解耦的跨组件通信. 然而, 这一灵活、广播式、扁平的机制带来了大量滥用、误用等问题. 组件的扁平化管理使任意组件之间都存在可能的交互, 复杂性随组件数量呈平方量级增长, 难以支撑开放动态的人机物融合应用场景.

在概念和方法学层面, 本文倡议在面对复杂应用需求时, 借鉴人类社会的组织形态, 将应用组织为树状结构, 下层节点接收来自上层节点的指令, 并负责向上层节点汇报数据 (其所管辖子系统之状态). 如图 1 所示, 应用需求 (即用户对未来系统状态的预期) 被提交给根组件 (智能体), 根组件 (智能体) 则继续将需求分解给下层节点, 直到软件孪生世界中的每个终端节点 (一个软件, 代表一个传感器、一组 API 等) 对需求做出响应. 许多已经获得成功的系统都借鉴了这一设计思路, 例如已有五十年历史的 Actor Model, 至今仍是复杂软件系统设计中常用的范式.

在系统支撑层面, 泛在操作系统应负责在组件 (智能体) 之间向上传递流式的单点数据, 即面向人机物融合应用组件的新型语义 “管道”. 管道两侧组件之间传递的数据应具有明确的输入输出接口, 并由管道实现数据的同步、通信和流量控制. 在人机物融合应用场景中, 具有明确输入输出接口的管道能够实现分布式、冗余、不可靠的单点数据的归集和向上传递. 这样的设计利用了软件孪生下普遍冗余、互相印证的应用数据, 避免了对海量人机物融合应用数据使用可靠网络传输、分布式共识、持久化存储的高延迟消息队列的应用, 在上传下达的过程中对数据进行及时的抽象与舍弃, 以较少的开销实现人机物融合应用需求. 本文同样倡议泛在操作系统应在元级建立需求描述语言, 并将组件 (智能体) 与之能实现的需求进行关联, 实现自顶向下的需求分解.

4.3　基准设计: 智能原生

最后, 泛在应用数据时空交错, 在上传下达的数据从传感器读取、决策由人工智能模型生成时, 软件从完全遵循编程语言形式语义的逻辑制品演变为了结果普遍具有不确定性 (uncertainty) 的概率系统. 人机物融合应用的这一特点给传统软件开发方法带来了重大挑战: 软件必须具备自动适应数据不一致性、数据错误、数据缺失的能力, 但这恰是形式化的编程语言所欠缺的, 也是软件工程领域长久悬而未决之挑战. 随着深度神经网络和大语言模型的诞生, 人工智能时代涌现的各类新技术为这一挑战的解决带来前所未有的机遇.

为破解泛在数据多模冲突矛盾、应用场景需求难用形式语言描述之挑战, 本文提出泛在操作系统应用组件间应采用规范的数据和命令接口, 使组件 (智能体) 中的相当一部分代码能由人工智能自动生成、智能体间的数据和控制指令在流转过程中可被人工智能模型处理, 实现面向需求的自适应软件系统.

在概念和方法学层面, 本文倡议为数据的上传和指令下达的通用机制和运行环境, 实现开放环境中组件的动态生命周期管理和协同, 开发者将泛在操作系统能够处理的 “标准组件” 接入系统, 即可自动服务应用需求. 为此, 我们分析了人机物融合事件驱动应用的特点, 提出泛在操作系统组件处理单点数据的四要素: 时间 (when)、地点 (where)、人物 (whom)、事件 (what), 拥有这 4 个要素的单点数据是进程间传递数据的基本单位, 这使得组件之间流转的数据自动具有 “可解释” 的语义, 并具备被人工智能模型进一步处理的能力. 统一的接口还使得每一个组件都有明确的输入输出规约 (均为单点数据), 从而组件也可作为代码自动生成的基本单元, 通过大语言模型实现低代码编程.

在系统支撑层面, 泛在操作系统的主要任务是在分布式的场景中调度云、边、端侧的各类人工智能模型. 人机物融合泛在应用中的各个组件对人工智能模型有不同类型的需求. 从需求 (自然语言) 到代码的生成任务通常在应用部署时一次生成, 一般延迟不敏感但需要较大算力; 用于单点数据处理的模型的处理效率则关系到上传下达网络中顶层节点获知系统状态的延迟, 对其延迟和吞吐量均有较高需求, 但通常使用的模型也较为轻量; 用于协调决策的模型, 则对安全性有额外需求. 本文发掘了学件在泛在计算中的应用潜力, 并提倡在人机物融合应用中按需使用学件.

综上所述, 本文所提出的泛在操作系统基于软件孪生实现人机物融合泛在异构资源的抽象, 通过泛在智能体的逐层构建实现用户需求自顶向下分解, 智能体之间通过 “上传下达” 的有序协作解决复杂问题. 相比传统的基于组件的软件系统 [40] 和多智能体系统 [41], 智能原生的泛在智能体具备调用学件或大语言模型进行规划, 按需生成代码及调用工具逐步完成给定目标的能力, 从而可支撑开放环境

中更加复杂多样的人机物融合泛在应用.

5　泛在操作系统的应用案例

泛在操作系统通过资源抽象与复杂性分解的智能组件协作, 可以有效支撑多样化的泛在计算应用系统, 同时赋予泛在应用生态灵活扩展的空间. 此处通过操作系统智能助手、“天网” 监控工程、疾病预防与控制系统 3 个应用案例展示泛在操作系统支撑下的案例系统设计及生态扩展优势.

5.1　泛在计算场景下的操作系统智能助手

背景与动机. 在人工智能模型爆发式发展的今天, 国内外产业界对操作系统智能化的未来趋势有所共识: Microsoft, Google, Apple 等前沿科技公司相继推出或规划了深度集成智能助手 (OS copilot) 的新型操作系统 [42~44], 致力于实现个人计算设备上跨应用的智能协同, 通过统一的自然语言接口控制异构软硬件资源. 用户可以通过语音对话向系统搭载的智能助手发布命令, 智能助手将判断出与用户需求相关的应用, 转发命令并收集结果. 因为构建在操作系统内部, 智能助手不仅有潜力处理跨应用的用户需求, 还能构建本地用户画像, 提供高度个性化的智能服务. 与此同时, 国产操作系统也在加速智能化的进程 [45, 46]. 然而, 现有的操作系统智能助手存在以下局限性.

• 现有技术提供的智能服务局限于特定范围内的系统原生应用, 额外定制其他可被智能助手控制和管理的应用需要付出高昂的开发成本, 难以应对泛在计算环境下多样化的用户需求. 此外, 目前跨应用服务也局限于系统原生对特定智能应用的增强, 缺乏可扩展性, 未能释放操作系统智能助手的全部潜力.

• 现有技术仅实现了基础的智能服务管理功能, 缺乏完善的底层机制来整合人机物融合应用的异构资源. 在泛在计算环境下, 其构建的用户画像具有局限性, 难以提供精准的个性化智能服务.

• 由于终端应用的智能化进程尚未普及, 自然语言交互仍处于辅助地位, 无法取代传统的图形界面, 命令行终端及键盘、鼠标等交互方式.

系统设计. 元级化软件定义的泛在操作系统能够为新型操作系统智能助手的构建、部署和运行提供理想的平台, 为国产操作系统提供 “弯道超车” 的契机. 典型的操作系统智能助手实例如图 2 所示. 首先, 得益于上传下达的基本形态, 新型智能助手提供的服务将跨越单一应用的能力边界. 在管道 – 组件的架构下, 复杂的智能应用被拆解为功能明确, 职责单一的组件, 作为底层数据源通过管道向上传递各类信息和资源. 而系统在面对复杂的场景级用户需求时, 能够通过管道将划分好的子任务下达给各个组件, 获取指定的资源以完成特定的任务, 例如从手机上的不同应用中分别自动提取最近一周的交通出行, 用餐消费、便签、照片、天气等, 自动形成一份逻辑清晰的旅途日记, 这对于个体智能应用来说是难以实现的.

其次, 软件孪生的基础能力使新型智能助手可以提供能力更强、覆盖范畴更广的智能服务. 一方面, 随时间演化的全系统状态能为组件提供更加丰富而广泛的信息和资源, 智能助手能够根据智能应用的具体功能筛选有效信息并进行相应决策; 进一步地, 智能助手还能通过对外部输入资源和系统内用户历史行为的统合构建精准的用户画像, 提供高度个性化的智能服务. 另一方面, 泛在操作系统将一切人机物融合异构资源视为软件, 这一高层抽象使其能够屏蔽环境的不稳定性和应用的复杂性, 高效地构建泛在计算场景下的智能应用, 而不再局限于传统软件的范畴. 基于上述两点, 泛在操作系统所提供的智能服务能够实现功能上的全面覆盖, 即用户可以通过自然语言完成任何系统行为. 因此, 自然语言将取代传统接口成为主要的交互方式, 用户可以通过智能助手观测、控制和管理系统中一切智能应用的所有行为.

最后, 智能原生的基准设计支撑了泛在智能应用的自适应开发和演化. 泛在操作系统组件间的接口具有规范性, 而组件的单点数据处理语义使得组件间传递的数据具有 “可解释” 的语义, 这种明确且

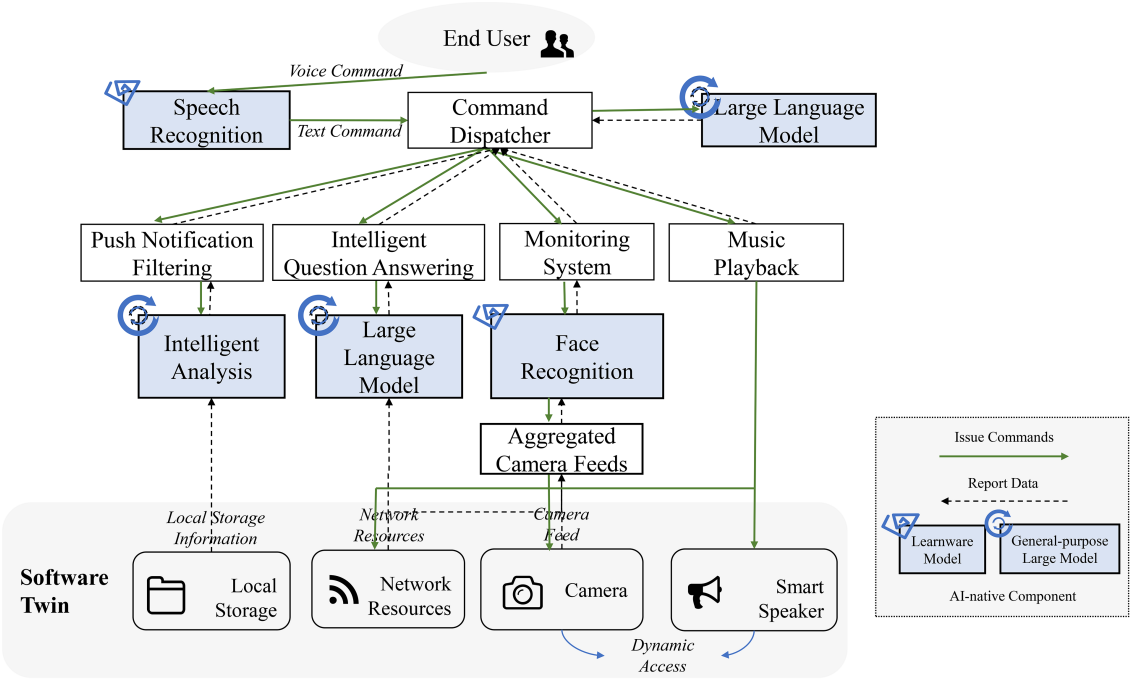

图 2　(网络版彩图) 操作系统智能助手工作实例.

Figure 2　(Color online) Operational example of an intelligent assistant for operating systems.

易于被人工智能模型理解和处理的规约使得大语言模型能够以组件为单位自动生成代码, 显著降低开发新型智能助手所需的编程代价. 该设计也让组件具备动态接入和故障容错的能力, 有效支撑新型智能助手的自适应演化. 此外, 泛在操作系统能够调度云、边、端侧的各类人工智能模型来满足智能助手所依赖的各类泛在应用的多样化需求. 具体地讲, 在设计和实现操作系统智能助手时, 能够根据不同智能应用对延迟、吞吐量、算力, 以及模型特长的不同要求为其分配合适的人工智能模型. 在泛在计算场景下, 诸如通用大语言模型和学件等各类人工智能模型均被设计为具备统一规范的组件, 使复杂环境下人工智能模型的高效调度和协调成为可能.

生态扩展. 泛在操作系统的设计理念天然地为新型操作系统智能助手提供了软件生态演化和扩展的系统和环境支撑. 上传下达的类社会组织形态高效地分解了应用需求和软件系统的复杂性, 规范的组件间数据和命令接口为泛在应用组件赋予了高可复用性和可扩展性. 一方面, 新型智能助手自身易于成长、演化和扩展. 统一和规范的组件接口保证了智能助手内部实现的低耦合与高度模块化, 显著减少了编程代价和维护代价. 例如对于受智能助手管理的监控系统和相册智能管理应用, 人脸识别和照片解析应当共享图像识别的基础功能, 这在泛在操作系统中将以单个组件被共享的形式存在, 而不需要由不同厂商提供的不同智能应用各自开发相同的功能.

另一方面, 新型智能助手是泛在场景下软件生态的一部分. 泛在计算场景下的诸多人机物融合异构资源覆盖了广阔的应用需求范围, 必定具备数量不可忽视的共通功能. 在泛在操作系统的管理下, 任何接入生态的新资源、新应用都以一系列解耦的组件的形式向智能助手开放: 如果需要智能助手支持一些新接入的应用, 相当部分的所需代码都应当已经被该应用的某些组件所实现, 可以轻松复用. 例如对某些新型视频流智能处理应用, 开发者可以复用生态中已有的流媒体解析、摄像头管理、人工智能模型交互等组件 (即已经被智能助手所管理的其他同类应用依赖的组件), 只需要实现新应用独有的一些核心组件, 例如应用前端界面、新颖的智能化功能、人工智能模型的具体利用策略等. 从传统软件的视角看, 这意味着智能助手能够从软件栈的任意层级的任意功能模块中获取其所需的信息和资源, 相比传统场景下操作系统应用模块化具有更高的灵活性和可扩展性.

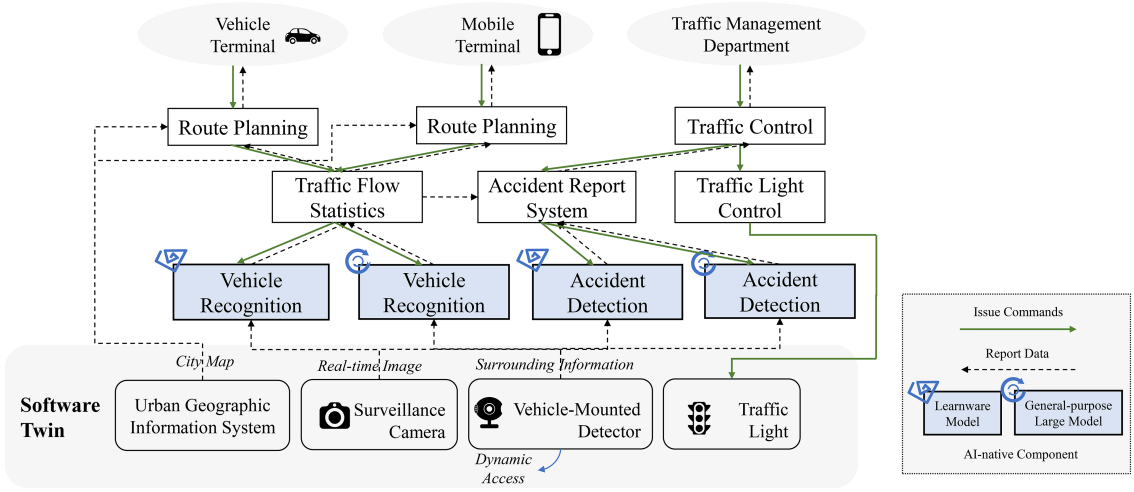

图 3　(网络版彩图) “天网” 系统在交通管理中的应用.

Figure 3　(Color online) Application of the “Skynet” system in traffic management.

5.2　泛在操作系统赋能 “天网” 监控工程

背景与动机. “天网监控系统” 是一个由公共道路和相关区域部署的传感器、摄像头等设备组成的城市安防监控工程, 旨在为数字化社会的建设提供重要的资源基础. 该系统通过在城市各处布置的摄像头和传感器, 整合了城市中的所有监控信息, 能够实时监测和分析城市的安全状况, 保障公共安全和社会稳定. 在当前人机物融合的背景下, 天网系统的作用将不再局限于传统的安防监控, 还可以扩展到智慧城市建设、交通管理、环境监测等多个领域, 这对新一代天网提出了全新的要求.

• 数据来源兼容性. 随着各类 IoT 设备的接入和涌现, 天网系统需要适配各类设备、各种系统. 除了已有的各类摄像头, 环境中的每个 IoT 设备都可以成为数据源, 但不同的设备使用不同的系统, 产生的信息具有不同的格式, 这给天网系统的进一步迭代开发带来了巨大的困难.

• 终端应用多样性. 天网获得信息可用于智慧城市建设的各个方面, 例如社会人员流动、智能交通规划、环境监测等, 所以需要天网系统具备良好的支持应用开发的能力. 天网系统需要提供良好的数据抽象, 易于使用的分布式设计, 以及便利的模块化设计.

• 数据处理复杂性. 由于各类设备的接入以及终端应用的不同需求, 需要天网系统能够对数据进行各种处理如数据一致性分析、数据清洗等.

系统设计. 泛在操作系统能够有效支撑天网系统的未来资源整合和系统架构的迭代更新. 典型的应用架构如图 3 所示. 首先, 泛在操作系统软件孪生的特点契合城市安防监控、智能规划等应用的需求. 城市的动态变化对应了泛在操作系统的全系统状态的变化, 因此泛在操作系统能够为天网系统提供全局视角以及决策下发的通道, 以应用需求的视角关注城市的发展变化并作出应对, 使得新应用的开发只需要通过软件定义所需数据与数据的处理方式, 无需关注具体设备, 降低了组件之间的数据传输难度, 这一特点能降低新应用开发的难度.

其次, 泛在操作系统上传下达的管道 – 组件架构能够切分终端应用, 提高开发效率. 以 Spring [47], Vue [48] 等框架为代表的模块化设计是当今软件开发必不可少的思想, 泛在操作系统通过管道 – 组件架构, 分解终端应用: 数据在应用的各个组件之间流动, 由泛在操作系统根据应用描述进行管理, 组件可以复用, 提高开发效率; 同时, 泛在操作系统通过下达的方式对数据进行控制, 对数据的约束发出后由泛在操作系统传达生效. 以智慧交通为例, 泛在操作系统可以通过各个路段的摄像头实时监测交通流量. 摄像头能够计数路段上的车辆数量, 并结合地图信息分析车流分布, 从而实现对交通状况的智能化管理. 当某个路段出现拥堵时, 系统可以自动广播信息, 与车机等设备联动, 为驾驶员动态调整路

线选择, 同时可以与交通部门协作, 限制拥堵路段流量.

最后, 统计的全局信息可以进一步为未来的道路建设与改造提供参考. 在上述应用中, 智慧交通应用可以复用摄像头监控、路线规划等应用组件, 通过命令下达方式过滤数据, 获取的信息可以进一步被其他应用复用. 此外, 泛在操作系统 AI 内嵌的特性能够帮助组件自动生成, 向应用提供智能接口, 进一步提高开发效率, 满足未来天网终端应用多样性的要求.

生态扩展. 泛在操作系统为天网系统提供了丰富的扩展能力. 泛在操作系统通过规范化的数据接口, 实现了数据传输与数据处理的解耦, 从而为应用的开发与扩展提供了极大的灵活性. 以交通管理系统为例, 多个节点可以复用泛在操作系统中已有的数据处理组件, 如摄像头数据处理、图像识别、路径规划等模块. 这种模块化设计不仅简化了开发流程, 还显著提高了系统的可维护性和可扩展性. 随着未来更多数据源的接入, 如公共交通系统中的人流量探测设备, 泛在操作系统的生态优势将进一步凸显. 开发者只需复用现有的数据处理模块, 即可快速将这些新数据源整合到交通管理应用中, 从而实现应用的快速迭代与演化.

此外, 泛在操作系统的生态还支持应用之间的深度集成. 例如, 随着新能源汽车的普及, 充电桩资源的使用愈发紧张. 通过将交通管理应用中的组件和数据复用, 交通流量等信息可以通过管道连接到充电桩规划或管理应用中, 从而优化充电桩的布局和使用效率. 这种基于泛在操作系统的组件复用机制, 不仅为应用开发提供了极大的灵活性, 还为系统的更新和维护带来了便捷. 因此, 泛在操作系统的引入为天网监控系统的未来发展提供了强有力的支持. 它能够有效管理生态中异构的软硬件资源, 支持不同应用开发过程中资源组件共享和复用, 为泛在应用提供灵活性和可扩展性. 这不仅进一步提升了城市的安全管理和资源调度能力, 还能够支撑现代智能化城市中不断涌现的新需求, 为智慧城市的建设奠定坚实基础.

5.3　基于泛在操作系统的疾病预防与控制系统

背景与动机. COVID-19 的全球大流行对各国的公共卫生系统造成了极大的冲击. 由于缺乏高效的疫情预警系统, 许多国家在疫情初期未能准确捕捉传染病传播的早期信号, 导致疫情防控滞后, 错失了阻止病毒扩散的关键时机. 生产生活和社会经济因此受到深远影响. 在万物互联的时代, 海量数据的支持可以大幅提升疾病防控系统的效率. 该系统通过收集分析各类健康数据, 识别传染病大流行的早期信号, 预测疫情的潜在暴发, 从而帮助公共卫生部门、医疗机构、企业和公众及时采取预防措施, 降低疫情扩散风险, 对于国民生命健康的保障具有重要意义.

世界各国及相关国际组织均设有公共卫生部门, 负责传染类疾病的监测与管理. 美国疾病控制与预防中心 (CDC) 是负责公共卫生监测、疾病预防与控制、健康教育的联邦机构. 为了提高疫情监测的效率, CDC 在 2021 年提出了 “北极星架构”, 旨在通过建立统一的数据格式和规范, 优化数据收集、整合和分析流程, 同时增强系统间的互操作性, 以实现更快速、精准的公共卫生决策和应对措施. 然而, 尽管 “北极星架构” 具有巨大的潜力, 目前却仍未完全开发和落地, 在数据整合、实时性与系统协同方面有待进一步完善. 与此同时, 我国正在逐步建设多级公共卫生监测体系, 探索引入数字化、智能化的监测手段, 以提升传染病的监测与管理效率.

理想的疾病预防与控制系统不仅能整合来自多源的异构数据, 还能实现高效的信息传递和精准的疫情预测, 以便公共卫生部门最大限度地降低疫情传播风险. 然而, 实现一个高效、可靠的疾病预防与控制系统面临诸多挑战.

- 设备碎片化. 不同传感器和监测设备的数据格式、标准不统一, 导致疫情数据难以整合和共享, 公共卫生系统难以快速、准确地获得全面数据, 延误了潜在疫情风险的识别和处理.
- 传输效率低下. 异构的软件和硬件环境导致数据在 “端 – 运营商 – 云” 架构中传输效率低, 影响实时监测的准确性和数据的可靠性.
- 分析处理滞后. 传统系统依赖人工或低效算法来识别异常信号, 限制了疫情预警系统的响应速

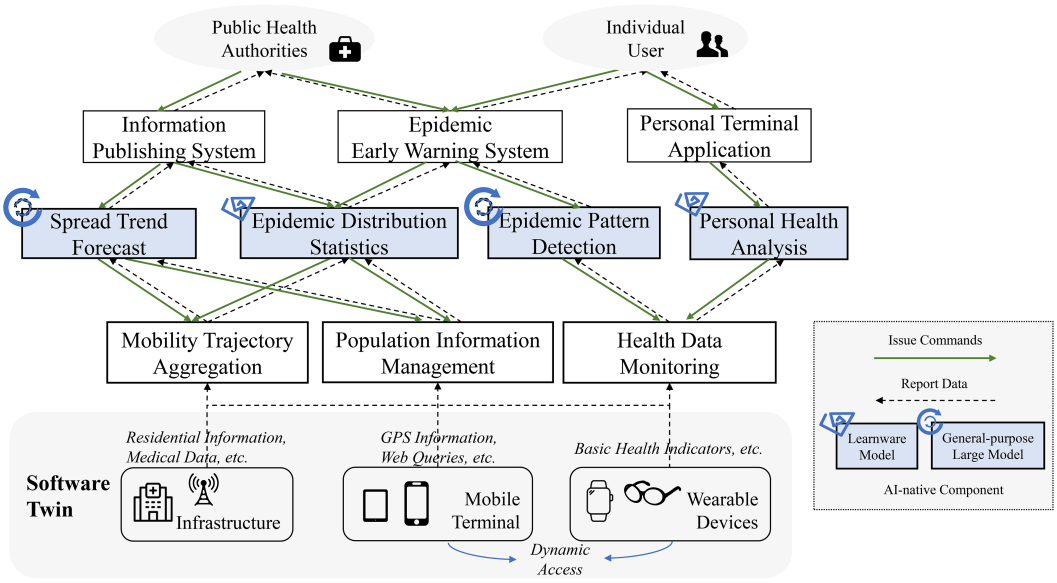

图 4　(网络版彩图) 基于泛在操作系统的疾病防控系统.

Figure 4　(Color online) Disease prevention and control system based on ubiquitous operating system.

度, 影响其动态调整、预测和决策支持能力.

系统设计. 基于泛在操作系统的疾病预防与控制系统能够有效应对数据碎片化、传输效率低下以及实时分析能力不足等问题, 从而提升公共卫生系统在疫情监测与管理中的整体表现. 系统的典型架构如图 4 所示. 泛在操作系统的软件孪生理念提供了一个全景视图, 使多来源数据 (包括医疗记录、定位信息、健康监测数据等) 能够实时整合、同步并展示, 为各级公共卫生管理部门提供统一的信息支持. 系统通过构建传播趋势预测、疫情分布统计和疫情模式检测等多模块, 建立了疫情态势等动态监控机制. 不同数据源的差异性被自动屏蔽, 所有疫情相关信息均在统一的全景视图中呈现. 公共卫生部门和医疗机构可以基于这一全面的数据视图, 迅速了解疫情进展, 并据此制定和调整预防措施.

泛在操作系统的 “上传下达” 机制为疾病预防与控制系统建立了分层、结构化的数据传输通道, 大幅提升了数据传输的效率与精准度. 在数据上传过程中, 基础设施、移动终端和可穿戴设备等负责采集用户健康数据, 包括 GPS 定位信息、健康监测数据及居住信息等. 这些数据经逐级汇聚, 最终传送至中央控制系统, 为疫情分析和趋势预测提供实时数据支持. 在指令下达过程中, 中央控制系统根据汇总的数据分析的结果, 生成应对措施, 通过信息发布系统和疫情预警系统逐级传达至各模块、部门和个人终端应用, 实现精准的疫情防控.

此外, 泛在操作系统的智能原生设计使得疾病预防与控制系统在应对突发疫情时具备高度自适应性和拓展能力. 通过标准化接口, 系统能够在云、边、端设备中灵活调度人工智能模型对实时数据进行自动化处理, 识别趋势并预测疫情暴发. 借助 AI 代码生成技术, 系统可动态接入新的功能模块, 实现自动化代码生成与集成, 无需进行繁琐的手动开发. 疫情分析模块结合健康数据监控与人群信息管理, 使疫情相关的决策方案在智能模型的支持下实时生成并调整, 为公共卫生部门提供高效、灵活的决策支持.

生态扩展. 泛在操作系统赋予疾病预防和控制系统强大的生态扩展能力, 依托规范化的组件接口和统一的数据传输标准, 泛在操作系统应用天然具备跨软件共通的特性. 这种设计不仅支持系统的快速功能扩展, 还在公共卫生管理中形成了开放、共享的技术生态. 以医院管理系统为例, 该系统可以通过共享组件与疾病预防和控制系统紧密协作. 当医院管理系统记录患者就诊信息时, 其内置的健康数据组件会实时收集患者的体温、血压、诊断结果等数据. 这些数据通过标准化接口直接上传到疾病

预防与控制系统等数据库中, 无需额外的格式转换或协议对接, 从而大幅提升数据采集与传输的效率. 与此同时, 医院系统还可以调用疾病预防系统中已有的疫情态势分析组件, 将区域内感染病例的趋势图和风险评估结果集成到其内部系统中, 帮助医护人员更全面地了解本地疫情情况, 优化就诊流程和应急预案. 此外, 在流行性传染病的防控方面, 泛在操作系统生态展现出卓越的适应能力. 一旦检测到传染病流行的初期信号, 疾病预防与控制系统会通过标准化接口, 迅速在生态内部同步所有相关参数, 使包括疾病预防与控制系统在内的相关应用都能及时适配这一变化: 医院管理系统可立即接入这些更新的组件, 调整筛查标准, 增加相关监测项目; 社区卫生平台也可以利用整合后的数据开展区域筛查, 动态生成风险热力图, 提示高风险区域的人群加强防护. 这种高效的功能拓展能力确保了生态内各系统能够快速响应新需求, 为公共卫生部门在疫情中的实时决策提供了有力支持.

6　总结与展望

物联网产业的大规模扩张带来了泛在计算的新时代, 但泛在环境的复杂性、业务需求的多样性、高度受限的计算资源为人机物融合应用开发带来了挑战; 同时, 人工智能技术的高速发展也为人机物融合泛在应用的智能化带来了全新的机遇和挑战. 为此, 本文提出了元级化软件定义的泛在操作系统, 认为人机物融合应用构建的元级化方法包含三大主要理念: 在上帝视角观测应用场景的基础能力, 拟人类社会组织方式的应用基本形态, 以及面向人工智能模型的基准设计, 分别解决异构资源抽象、应用需求分解和智能原生支持. 本文分析了操作系统智能助手、“天网” 监控系统和疾病防控 3 个典型的泛在操作系统应用案例, 阐述了泛在操作系统应对真实世界人机物融合应用的开发需求与挑战的能力.

目前, 我们已在人机物融合泛在计算的多个关键支撑技术方面取得进展. 在开放环境可靠感知方面, 通过多维度 (增量、并发、混合式等) 约束检测技术 [49~51] 实现对毫秒级开放环境场景上下文数据的一致性高效检测, 相较基准工作效率提升两个数量级 [51], 能够有效支撑人机物融合场景下泛在数据的高效质量保障; 在领域软件自动生成方面, 通过领域特定抽象实现软件复杂性的简化, 结合智能化手段实现对领域关键软件或组件的自动生成, 如安卓应用跨设备版本生成 (成功率超过 80%) [52]、编译器测试关键变异算子生成 [53] 等, 能够促进泛在应用组件的智能原生研究; 在编程语言支撑方面, 提出完全自主可控的软件分析系统太阿 [54], 广泛支持编程开发各类典型程序分析需求 (控制流分析、指针分析等), 相较业界主流分析工具 Soot [55] 分析效果相当甚至更优, 为泛在应用开发提供可用易用的基础分析支撑.

本文着重探讨了泛在操作系统的元级化理论及其落地应用的能力. 未来, 我们将开发工具以支持已有软件资源的泛在化改造, 并进一步探索泛在操作系统在数据隐私和网络安全防护等方面的设计理念, 以及高性能传输与通信, 数据一致性验证等实际问题的解决方案, 为人工智能时代的人机物融合泛在应用提供更可靠的系统支撑.

参考文献

1　Weiser M. The computer for the 21st century. SIGMOBILE Mob Comput Commun Rev, 1999, 3: 3–11

2　Mei H, Guo Y. Network-oriented operating systems: status and challenges. Sci Sin Inform, 2013, 43: 303–321 [梅宏, 郭耀. 面向网络的操作系统 —— 现状和挑战. 中国科学: 信息科学, 2013, 43: 303–321]

3　Mei H, Guo Y. Toward ubiquitous operating systems: a software-defined perspective. Computer, 2018, 51: 50–56

4　Liu Z, Wang J. Human-cyber-physical systems: concepts, challenges, and research opportunities. Front Inform Technol Electron Eng, 2020, 21: 1535–1553

5　曹东刚, 邬惠峰. 面向工业物联网的泛在操作系统. 中国计算机学会通讯, 2023, 19: 23–28

6　Mei H, Cao D G, Xie T. Ubiquitous operating system: toward the blue ocean of human-cyber-physical ternary ubiquitous computing. Bull Chinese Acad Sci, 2022, 37: 30–37 [梅宏, 曹东刚, 谢涛. 泛在操作系统: 面向人机物融

合泛在计算的新蓝海. 中国科学院院刊, 2022, 37: 30–37]
7 Huang Z, Shen Y, Li J, et al. A survey on AI-driven digital twins in Industry 4.0: smart manufacturing and advanced robotics. Sensors, 2021, 21: 6340
8 Ryan M, Isakhanyan G, Tekinerdogan B. An interdisciplinary approach to artificial intelligence in agriculture. NJAS-Impact Agric Life Sci, 2023, 95: 2168568
9 Tjoa E, Guan C. A survey on explainable artificial intelligence (XAI): toward medical XAI. IEEE Trans Neural Netw Learn Syst, 2021, 32: 4793–4813
10 Guttman R H, Moukas A G, Maes P. Agent-mediated electronic commerce: a survey. Knowl Eng Rev, 1998, 13: 147–159
11 Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: deep learning. Genet Program Evolvable Mach, 2018, 19: 305–307
12 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
13 Mahesh B. Machine learning algorithms — a review. Int J Sci Res, 2020, 9: 381–386
14 Bottou L, Curtis F E, Nocedal J. Optimization methods for large-scale machine learning. SIAM Rev, 2018, 60: 223–311
15 Liu J, Shen Z, He Y, et al. Towards out-of-distribution generalization: a survey. 2021. ArXiv:2108.13624
16 Yang J, Zhou K, Li Y, et al. Generalized out-of-distribution detection: a survey. Int J Comput Vis, 2024, 132: 5635–5662
17 Zhou Z H. Learnware: on the future of machine learning. Front Comput Sci, 2016, 10: 589–590
18 Bell G. Bell's law for the birth and death of computer classes. Commun ACM, 2008, 51: 86–94
19 Pagano F, Verderame L, Merlo A. Understanding Fuchsia security. 2021. ArXiv:2108.04183
20 Coulouris G, Dollimore J, Kindberg T, et al. Distributed Systems: Concepts and Design. Reading: Addison-Wesley, 2011
21 Ghemawat S, Gobioff H, Leung S T. The Google file system. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), 2003. 29–43
22 Hunt P, Konar M, Junqueira F P, et al. ZooKeeper: wait-free coordination for Internet-scale systems. In: Proceedings of the USENIX Annual Technical Conference (USENIX ATC), 2010. 145–158
23 Pike R, Presotto D, Thompson K, et al. Plan 9 from Bell Labs. Comput Syst, 1995, 8: 221–254
24 Barham P, Dragovic B, Fraser K, et al. Xen and the art of virtualization. In: Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), 2003. 164–177
25 Barroso L A, Clidaras J, Hölzle U. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. 2nd ed. Berlin: Springer, 2013
26 Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Commun ACM, 2010, 53: 50–58
27 Amazon Inc. Cloud computing services — Amazon Web Services (AWS). 2024. https://aws.amazon.com/
28 阿里巴巴集团. 阿里云 —— 全球云计算服务及云解决方案提供商, 2024. https://www.alibabacloud.com/
29 曹春, 马晓星. 面向人机物融合应用的场景计算机. 中国计算机学会通讯, 2020, 16: 31–35
30 Naqvi S N Z, Yfantidou S, Zimányi E. Time series databases and InfluxDB. Studienarbeit, Université Libre de Bruxelles, 2017, 12: 1–44
31 Ntalampiras S, Arsic D, Stormer A, et al. Prometheus database: a multimodal corpus for research on modeling and interpreting human behavior. In: Proceedings of the 16th International Conference on Digital Signal Processing (ICDSP), 2009. 1–8
32 Wang C, Qiao J, Huang X, et al. Apache IoTDB: a time series database for IoT applications. Proc ACM Manag Data, 2023, 1: 1–27
33 Shvachko K, Kuang H, Radia S, et al. The Hadoop distributed file system. In: Proceedings of the 26th Symposium on Mass Storage Systems and Technologies (MSST), 2010. 1–10
34 Russel S, Norvig P. Artificial Intelligence: A Modern Approach. London: Pearson, 1995
35 Rao A S, Georgeff M P. BDI agents: from theory to practice. In: Proceedings of the 1st International Conference on Multiagent Systems (ICMAS), 1995. 312–319
36 Wooldridge M, Jennings N R. Intelligent agents: theory and practice. Knowl Eng Rev, 1995, 10: 115–152
37 Wooldridge M. An Introduction to Multiagent Systems. 2nd ed. Hoboken: Wiley, 2009
38 Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents. Front Comput Sci, 2024,

18: 186345

39 Katsifodimos A, Schelter S. Apache Flink: stream analytics at scale. In: Proceedings of the IEEE International Conference on Cloud Engineering Workshop (IC2EW), 2016. 193–193

40 Heineman G, Councill W. Component-based Software Engineering: Putting the Pieces Together. Reading: Addison-Wesley, 2001

41 Shoham Y, Leyton-Brow K. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge: Cambridge University Press, 2008

42 Microsoft. Windows Copilot. 2024. https://www.microsoft.com/en-us/microsoft-copilot/personal-ai-assistant

43 Apple Inc. Apple intelligence. 2024. https://www.apple.com/hk/en/apple-intelligence

44 Google. Gemini App — a new AI assistant on your phone. 2024. https://gemini.google.com/app/download

45 Wuhan Deepin Technology. Deepin: 基于 Linux 的开源国产操作系统. 2024. https://www.deepin.org/

46 OpenKylin. 开放麒麟社区. 2024. https://www.openkylin.top/

47 Arthur J, Azadegan S. Spring framework for rapid open source J2EE Web application development: a case study. In: Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and the 1st ACIS International Workshop on Self-Assembling Wireless Network (SNPD/SAWN), 2005. 90–95

48 You E. Vue.js — the progressive JavaScript framework. 2024. https://vuejs.org/

49 Wang H, Xu C, Guo B, et al. Generic adaptive scheduling for efficient context inconsistency detection. IEEE Trans Softw Eng, 2019, 47: 464–497

50 Xu C, Cheung S C, Chan W K, et al. Partial constraint checking for context consistency in pervasive computing. ACM Trans Softw Eng Methodol, 2010, 19: 1–61

51 Zhang L, Wang H, Xu C, et al. INFuse: towards efficient context consistency by incremental-concurrent check fusion. In: Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 2022. 187–198

52 Li C, Jiang Y, Xu C. Push-button synthesis of watch companions for Android APPs. In: Proceedings of the 44th International Conference on Software Engineering (ICSE), 2022. 1793–1804

53 Ou X, Li C, Jiang Y, et al. The mutators reloaded: fuzzing compilers with large language model generated mutation operators. In: Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2024

54 Tan T, Li Y. Tai-e: a developer-friendly static analysis framework for Java by harnessing the good designs of classics. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), 2023. 1093–1105

55 Vallee-Rai R, Co P, Gagnon E, et al. Soot — a Java bytecode optimization framework. In: Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON), 1999. 13–23

System support for ubiquitous human-cyber-physical fusion applications

Yunfan CAO[1,2], Chaoyi ZHAO[1,2], Hanzhi LIU[1,2], Jiayi WANG[1,2], Huiyan WANG[1,2], Ping YU[1,2], Chun CAO[1,2], Chang XU[1,2], Xiaoxing MA[1,2] & Yanyan JIANG[1,2*]

1. *State Key Laboratory for Novel Software Technology, Nanjing 210023, China*
2. *School of Computer Science, Nanjing University, Nanjing 210023, China*
* Corresponding author. E-mail: jyy@nju.edu.cn

Abstract Ubiquitous computing brings new possibilities for human-cyber-physical fusion applications, but it also faces the problems of heterogeneous resource management, diversity of requirements and limited computing resources. Meanwhile, the rapid development of artificial intelligence models presents both opportunities and challenges to the development of intelligent applications. To this end, this paper proposes the design concept of meta-level software-defined ubiquitous operating system, which introduces the "software twin" technology to realize the abstract management of heterogeneous resources, utilizes the "report and issuance" hierarchical structure to decompose complex requirements, and supports AI models through AI-native design. This paper further illustrates the practical advantages and potential value of this ubiquitous operating system design concept through three typical ubiquitous computing application scenarios: an intelligent OS assistant, a Skynet monitoring system, and a disease prevention and control system, highlighting the broad application prospects of human-cyber-physical fusion in the intelligent era.

Keywords ubiquitous computing, ubiquitous operating system, software engineering