信息高铁的低熵高通量性质验证

俞子舒 李奉治 郑守建 王一帆 张星洲 彭晓晖 徐志伟

(中国科学院计算技术研究所 北京 100190)

(中国科学院大学 北京 100049)

摘 要 针对“人机物”三元融合的万物智能互联时代需求，中国科学院计算技术研究所的学者提出了“信息高铁”高通量低熵算力网的设计构想，在基础设施层提供低熵有序的支持，从而显著提升性能与品质. 但信息高铁构想尚缺乏实验数据支撑. 本文量化定义了通量、良率、CPU 熵，分别用于刻画系统的性能、品质、无序程度；构建了信息高铁原型系统；并与基于 Kubernetes 容器编排的云计算系统进行对比. 通过北京-南京的广域实验，验证了信息高铁原型系统可数量级提升良率和通量，其正则化后的 CPU 熵降至 K8s 系统的 50%以下. 本文通过提供实验数据分析结果，初步验证了信息高铁的低熵高通量潜力.

关键词 云计算；算力网；低熵计算；高通量计算；占而不用

中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2023.02302

Can Information Superbahn Achieve Low Entropy and High Goodput? A Validation Study

YU Zi-Shu LI Feng-Zhi ZHENG Shou-Jian WANG Yi-Fan ZHANG Xing-Zhou
PENG Xiao-Hui XU Zhi-Wei

(*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190)

(*University of Chinese Academy of Sciences, Beijing* 100049)

Abstract We are entering a new era of the Intelligent Internet of Everything, characterized by human-cyber-physical ternary computing. This requires information infrastructure with massive high-concurrency processing and high-quality user experience delivering capability. To address this challenge, a new type of cyberinfrastructure called Information Superbahn (ISB) has recently been proposed. ISB is envisioned as a planet-scale, low-entropy, high-goodput network computing system that aims to mitigate the negative impact of various disorderly factors on the user experience and significantly enhance system goodput and efficiency. However, experimental evidence is yet to be provided to support this vision. The purpose of this work is to provide such evidence. To achieve this, we build an ISB prototype system (ISB prototype) and compare its performance to that of a Kubernetes-based cloud computing system (K8s system) using the same workload and raw resources. This paper focuses on verifying the following three hypotheses. High-goodput hypothesis: ISB prototype should achieve orders of magnitude improvement in yield and goodput

收稿日期：2022-07-27；在线发布日期：2023-03-24. 本课题得到国家自然科学基金（No. 62072434, No.U19B2024）、中国科学院计算技术研究所创新课题（No. E261010）资助. **俞子舒**，博士研究生，中国计算机学会（CCF）学生会员，主要研究领域为分布式系统、运行时管理. E-mail: yuzishu19s@ict.ac.cn. **李奉治**，博士研究生，中国计算机学会（CCF）学生会员，主要研究领域为分布式系统、因果科学. **郑守建**，硕士研究生，中国计算机学会（CCF）学生会员，主要研究领域为边缘计算、资源管理. **王一帆**，博士，助理研究员，中国计算机学会（CCF）会员，主要研究领域为分布式系统、资源管理. **张星洲**，博士，助理研究员，中国计算机学会（CCF）会员，主要研究领域为分布式系统、编程方法. **彭晓晖**（通信作者），博士，副研究员，中国计算机学会（CCF）高级会员，主要研究领域为分布式系统、操作系统. E-mail: pengxiaohui@ict.ac.cn. **徐志伟**，博士，研究员，中国计算机学会（CCF）会士，主要研究领域为计算机体系结构、分布式系统.

under high load; Low-entropy hypothesis: both the yield and single-core goodput are significantly negatively correlated with CPU entropy, and the CPU entropy of the ISB prototype can be reduced to less than 50% of that of the K8s system; Prioritization hypothesis: the ISB prototype provides prioritization capabilities at the infrastructure layer. A wide-area experiment is designed to verify the three hypotheses. The experiment has two phases. In the first phase, we design a distributed intelligent workload to compare the two systems' goodput and yield. The workload involves edge data collection and cloud-based computation, featuring both burst and gradual workload changes. In the second phase, a red packet task is designed to validate the prioritization hypothesis. During this phase, the highest-priority tasks are required to complete first and execute exclusively, while tasks with the second-highest priority are required to finish earlier compared to most lowest-priority tasks. The ISB prototype is implemented based on one core idea and two key decisions. The core design idea is pooling all resources in the wide-area distributed system, then scheduling tasks in a logical queue based on their QoS requirements and priority labels, and transforming unordered task requests into well-ordered executions. To improve throughput and reduce system entropy, two key design decisions are adopted: the rapid scaling of wide-area shared resources and the native support of the DIP (Differentiation, Isolation, Prioritization) thesis. To accurately analyze the performance of both systems, this paper quantitatively defines and measures goodput, yield, and two types of CPU entropy. The results demonstrate that the above three hypotheses are verified. Firstly, compared to the K8s system, the ISB prototype improves yield by at least 1-fold and 1000-fold in 55.5% and 12.2% of the test samples, respectively, while goodput improves by at least 1-fold and 1000-fold in 58.9% and 14.4% of the test samples, respectively. Secondly, the ISB prototype's CPU entropy exhibits a significant negative correlation with yield and single core goodput, indicating that the low-entropy approach has the potential to improve the system's yield and throughput. In over 90% of the test samples, the regularized CPU entropy of the ISB prototype decreases to less than 50% of that of the K8s system. Lastly, the ISB prototype provides three levels of priority support at the infrastructure layer based on a wide-area synchronized clock, ensuring that high-priority tasks receive quality service. In summary, this work provides initial evidence indicating that Information Superbahn has the potential to significantly improve goodput over traditional cloud computing systems.

Keywords cloud computing; computility grid; low entropy computing; high-goodput computing; stranding

1 引 言

计算技术正走向人机物三元融合的智能万物互联时代，需要信息基础设施提供海量高并发的保质任务处理能力. 中国科学院计算技术研究所的学者提出了高通量低熵算力网（信息高铁）的设计构想. 信息高铁是一种提供低熵有序支持的新型信息基础设施，可以降低多种无序对用户体验的负面影响，并显著提升系统通量、系统效率和应用品质[1-3]. 它关注保质任务，即满足用户体验要求的任务，并追求两个核心性能指标. 一是通量（Goodput），即单位时间完成的保质任务数，它量化了系统性能；二是良率（Yield），即保质任务数占总任务数的比例，它量化了服务品质. 信息高铁旨在构建低熵有序的基础设施，为用户同时提供高通量的性能和高良率的品质.

但是，目前信息高铁的研究尚缺乏实证支持，还不能回答下述问题：

- 信息高铁的高通量目标能实现吗？能否在不增加资源的前提下数量级地提升通量？
- 信息高铁的高通量目标能够通过降熵实现吗？降什么熵？

本文的主要工作是通过量化分析，揭示现有互联网基础设施在高通量计算方面存在的不足，验证信息高铁具备数量级提升通量的潜力. 本文的主要创新成果是通过构造性实验，比较信息高铁原型系统（下称信息高铁原型）与基于 Kubernetes 的云计算实验系统（下称 K8s 系统），用实验数据初步回答

了上述问题. 上述两个系统使用相同数量的裸资源，处理相同的负载. 本文中信息高铁原型的任何提升或下降，皆是相对于 K8s 系统.

信息高铁原型以低熵有序的资源管理和使用为设计原则. 核心思路是将分布式系统的所有计算和存储资源看作一个整体，所有任务在一个逻辑队列中按照服务质量要求进行编排和调度，将无序的任务请求转变为有序的任务执行. 信息高铁原型的两个重要设计决定是基于跨域共享资源的快速扩容和原生提供 DIP[4]支持，以降低系统熵并提升通量.

本文主要贡献是：初步验证了通过低熵化的设计与实现，与现有 K8s 系统相比，信息高铁具备数量级提升通量的潜力.

其它贡献包括：

（1）量化定义了通量、良率和两种 CPU 熵，并在对比实验中采集了系统运行数据，并分析了这些指标 .

（2）通过构建原型系统，验证了信息高铁的高通量高良率低熵特性. 在高负载情况下，该系统的良率和通量提升了数倍到数千倍. 信息高铁原型的良率和单核通量均与 CPU 熵呈显著的负相关，正则化的 CPU 熵降至 K8s 系统的 50%以下.

（3）通过基于全局一致时钟的计算时空资源精细化管理和优先化支持，信息高铁原型有效缓解了由负载干扰、系统噪声等无序现象导致的性能干扰问题和 CPU 资源占而不用问题[4-6].

2 相关工作

信息高铁是为了应对智能万物互联时代新需求而提出的新型信息基础设施愿景[1-3]. 文献[3]总结了智能万物互联时代信息基础设施的 4 个较为鲜明的需求：支持“人机物”三元计算模式；提供高通量的计算性能；实现高品质的用户体验；提供全生命周期的应用效率. 面向万亿级信息设备产生的并发任务，信息高铁需要提供稳定的高通量处理能力. 针对用户体验，信息高铁需要在基础设施层提供支持，限制用户体验的无序波动，并在高负载下提供端到端延迟保证. 本文工作的主要目的是验证信息高铁同时具备提供高通量计算性能和高品质用户体验的潜力.

文献[2-3]介绍了信息高铁建设的指导思想. 从外部看，信息高铁是一台跨域的、统一控制的、灵活调度的分布式大电脑. 其通过一套分布式系统软件使得多组织、联邦共建的云网边端资源紧密协调配合，为用户提供易用的算力资源. 从内部看，信息高铁通过全链路多级多维度的测调、隔离等技术和机制，为用户提供低熵有序的资源共享，实现端到端延迟可控. 信息高铁还需具备较强的自适应性，满足用户对不同服务品质的需求. 就核心技术的路线而言，信息高铁应当将“李特尔定律”（Little’s Law）作为重要的设计指导原理，不应简单地将资源划分成上万个切片，分配给不同用户. 上述方式是走线路交换的老路，会导致平均延迟上升数个数量级[3]. DIP 猜想[4]指出，系统需要具备区分、隔离、优先化的能力，才可以限制系统无序带来的负面影响. 其中区分是指系统可自动区别访问资源的计算任务；隔离是指计算任务之间的计算时空不相交；优先化是指计算任务可以有不同的优先级，并由系统按优先级处理. 我们依据上述指导思想设计了信息高铁原型.

中科南京信息高铁研究院①建设了一个由北京、南京等多地的“端-边-云”开放实验室构成的信息高铁试验场，用于学术思想和核心技术的验证和原型系统的研发. 本工作使用了该试验场提供的部分计算、存储和网络资源.

本文使用 Kubernetes（K8s）作为信息高铁原型的对标系统. 它基于谷歌内部的 Borg[7]和 Omega 系统[8]改进实现，添加了自组虚拟网络、标签等[9]新机制. K8s 已经成为最广泛使用的云计算基础设施管理系统之一②，用来在集群上部署和编排容器，并提供自动扩缩容的功能. K8s 集群的自动扩缩容支持资源的弹性使用，但面对突发或快速变化的高并发请求时，难以保证系统的良率和通量. 受多种因素的影响，系统监测资源的状态变化只能精确到秒级，新的容器启动也需要数秒，且原容器中的任务请求无法在扩容后在基础设施层自动迁移到新容器. 因此，面对突增的高并发请求时，K8s 集群上的任务可能会请求超时或失败. 同时，在资源预配置阶段和缩扩容阶段，K8s 固有的分配不准确性和滞后性会带来资源占而不用的问题.

许多工作对 K8s 系统进行了性能评估. Víctor Medel 等[10]分析了 K8s 系统 Pod 的创建、结束时间，并使用 pov-ray 基准测试程序测量了 K8s 系统执行 CPU 密集型应用的开销（约 14%）. Vojdan Kjorveziroski 等[11]采用 14 个基准程序比较了不同 K8s 发

① 中科南京信息高铁研究院, http://www.ictnj.ac.cn/ 2023, 3, 21

② The past, present, and future of Kubernetes with Eric Brewer, https://cloud.google.com/blog/products/containers-kubernetes/the-rise-and-future-of-kubernetes-and-open-source-at-google 2021, 12, 8

行版（Kubespray, K3s 和 MicroK8s）的冷启动延迟、串行和并行性能，以及自动缩扩容性能，其结果显示针对资源受限设备进行优化的 K8s 发行版的冷启动延迟更低. 但是其进行自动缩扩容测试的负载变化频率为 5 分钟，没有测评更快速变化场景的性能. Changpeng Zhu 等[12]研究了在 K8s 上部署 spark 的性能并与裸机部署比较，最坏情况下性能损失达 89.3%. 本文只测量典型场景下 K8s 系统的性能，并与信息高铁原型比较.

云计算 3.0[5,13]期望支持无服务器计算（serverless computing）、智能计算等新兴负载，降低尾延时（tail latency）[14]和缓解占而不用（stranding）[5-6]问题，同时实现高品质用户体验和高资源利用率. 无服务器计算[15-16]是一种新的云计算执行模型，其使得开发者无需关心计算资源的配置、维护、管理等细节，只需要编写满足功能的代码，即可快速开发原型、投入生产并大规模应用[17]. “占而不用”指的是系统中存在被分配但是未被使用的资源，如阿里云 2018 年 CPU 平均利用率仅有 38%，资源预留严重影响了数据中心的资源效率[18]. 信息高铁原型系统的设计借鉴了无服务器计算的细粒度资源管理思路，通过更细粒度的资源按需分配和计算时空管理来提升原型系统的良率、通量和 CPU 的有效使用，进而验证信息高铁数量级提升通量的潜力.

超算和云计算中心等大规模算力系统需要多种新的度量方法来刻画服务质量和资源利用效率，以优化用户体验和运营成本. 谷歌使用尾延迟来度量计算中心的响应能力[14]. 文献[3]中指出信息高铁的性能指标是通量（保质任务吞吐率）和良率（保质任务比例），本文将量化定义通量和良率. 亚马逊云的 Cloudwatch①对 CPU 的使用率（CPUUtilization）进行了监测，我们结合服务质量的需求，进一步加入了 CPU 有效率的指标来衡量系统的 CPU 真实利用效率. SDCBench[19]使用延迟熵（latency entropy）来描述尾延迟变化，并衡量系统隔离能力和性能的不确定性，但是该指标不能衡量系统内部的无序程度. 本文提出并定义了 CPU 相关的两种熵，以衡量 CPU 使用和管理的无序性.

3　实验目标与指标定义

本文工作的重点是一个对比实验，其目标是验证以下三个假设（hypotheses）：

（1）高通量假设：在高负载下，信息高铁原型的良率和通量应有数量级提升；

（2）低熵假设：良率和单核通量均与 CPU 熵呈现明显负相关，同负载下信息高铁原型的 CPU 熵可降至 K8s 系统的 50%以下；

（3）优先化假设：信息高铁原型在基础设施层提供优先化[4]能力.

实验环境如图 1 所示，包括一套待测系统，即一套 K8s 系统，或一套信息高铁原型系统. 观察时段 $[t_{start}, t_{end}]$ 内系统的行为，即客户机与待测系统构成的端云系统的行为.

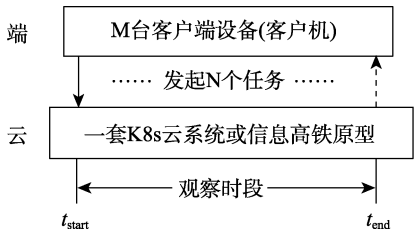


图 1　实验环境简图

文献[20]对一般情况下的保质任务、通量、良率、CPU 使用率和 CPU 有效率进行了定义。在此基础上，本文针对实际测量进行了补充和计算说明，并定义了两种 CPU 熵及其正则化。

表 1 给出了本文使用的符号及其说明.

表 1　记号及说明

记号	说明
G	系统通量
Y	系统良率
N 和 N_{QoS}	总任务数，保质任务数
U 、H_U 和 R_U	CPU 使用率、使用熵和正则化的使用熵
V 、H_V 和 R_V	CPU 有效率、有效熵和正则化的有效熵
T_{start} 和 T_{end}	发起首个任务时刻，收到最后一个结果时刻
T_{send} 和 T_{recv}	发起某任务的时刻，收到该任务结果时刻
T_{QoS}	任务保质要求，$T_{recv}-T_{send} \leqslant T_{QoS}$ 则保质

定义 1. 计算负载. 假设在观察时段内，客户机共发起 N 个任务. 客户机在 t_{start} 时刻发起第一个任务，所有 N 个任务中客户端接收到结果的最晚时刻为 t_{end}. 因此，观察时段是闭区间 $[t_{start}, t_{end}]$. 这 N 个任务的发起、执行、结果接收的全过程，构成了观察时段内的计算负载.

3.1　保质任务、良率与通量

定义 2. 保质任务. 图 1 中任一任务的行为如下：某个客户机在 t_{send} 时刻发起任务，待测系统执行任务，然后该客户机或另一台客户机在 t_{recv} 时刻接收到任务结果. 该任务的延时是 $t_{recv}-t_{send}$. 任务

① Cloudwatch, http://aws.amazon.com/cloudwatch 2023,3,21

保质要求为实验设定的时间量 T_{QoS}. 任务是保质的当且仅当 $T_{\mathrm{QoS}} \geqslant t_{\mathrm{recv}} - t_{\mathrm{send}}$.

定义 3. 良率和通量. 如图 1 所示, 在观测时间内, 假设客户端总共发起 N 个任务, 其中 N_{QoS} 个任务是保质任务, 则该观察时段 $[t_{\mathrm{start}}, t_{\mathrm{end}}]$ 内系统的良率（Y）和通量（G）分别定义为

$$Y = \frac{N_{QoS}}{N} \tag{1}$$

$$G = \frac{N_{QoS}}{t_{\mathrm{end}} - t_{\mathrm{start}}} = \frac{N \times Y}{t_{\mathrm{end}} - t_{\mathrm{start}}} \tag{2}$$

定义单核通量 = 通量 / 平均使用核数.

系统良率描述观测时间段内保质任务数占总任务数的比例. 良率越高, 用户体验越好. 系统通量描述单位时间内系统处理的保质任务数量. 通量越高, 系统性能越好.

3.2 CPU指标

本文使用 3 个 CPU 指标分别描述计算系统分配 CPU 资源情况、CPU 使用情况和执行了保质任务的 CPU 情况.

（1）分配核数: 系统实际分配给任务的核数. 在 K8s 等管理系统中, 常使用 CPU 配额核数来指导核数的分配. 配额核数（quota）是系统保证任务执行的最小可用核数.

（2）使用核数: 任务真实使用的核数. CPU 核只有被分配之后才可以被任务使用.

（3）有效核数: 执行保质任务使用的核数. 即, 如果任务保质, 则等于使用核数, 否则为 0.

定义 4. CPU 核数的两个比率. 在时刻 t, $alloc(t)$ 为系统总分配核数, $use(t)$ 为所有任务的总使用核数, $valid(t)$ 为执行保质任务的总有效核数. 可定义使用率 $U(t)$、有效率 $V(t)$ 两个比率:

$$U(t) = \begin{cases} 1, & use(t) = alloc(t) \\ \dfrac{use(t)}{alloc(t)}, & use(t) \neq alloc(t) \end{cases} \tag{3}$$

$$V(t) = \begin{cases} 1, & valid(t) = use(t) \\ \dfrac{valid(t)}{use(t)}, & valid(t) \neq use(t) \end{cases} \tag{4}$$

以及两个平均比率:

平均使用率 = 平均使用核数/平均分配核数

平均有效率 = 平均有效核数/平均使用核数

这些比率的定义有一些细节值得注意.

$alloc(t) \geqslant use(t) \geqslant valid(t) \geqslant 0$, 故使用率和有效率的取值范围是[0,1].

信息高铁原型无需预先分配 CPU 资源, 任务提交时调度器决定的所有任务配额核数之和即为系统总分配核数. K8s 系统中容器的 CPU 请求核数即为单个容器的 CPU 配额, 系统的总 CPU 配额为所有容器的 CPU 配额之和. 本文采用公式 5 估算 K8s 系统真实分配的核数. 假设 t 时刻, 总配额核数为 $quota(t)$, 则

$$alloc(t) = \max\left(quota(t), use(t)\right) \tag{5}$$

即取配额核数与实际使用核数中较大的为 K8s 系统真实分配的核数. 上述公式的依据是: 在 K8s 系统中, CPU 配额核数是 K8s 用于容器调度的重要参数, 当机器剩余配额大于 CPU 请求时才可能调度成功.

K8s 实验系统采用 Go 语言的协程机制实现, 故无法准确获取每一个请求的使用核数, 进而无法获取执行保质任务的总核数. 对于 K8s 系统可用如下方法估计 $vaild(t)$: 在 t 时刻, 所有正在执行的保质任务的平均使用核数之和. 保质任务的平均使用核数等于任务平均使用核时除以任务实际执行时长. 任务的平均使用核时采用实验的方式测得.

3.3 CPU资源管理的两个香农熵

在信息论中, 离散变量 X 的香农熵[21]定义为 $\sum_{i=1}^{n} p_i \times \log_2 p_i$, 其中 p_i 为变量 X 的第 i 个取值的概率. 对于连续随机变量 X, 可以通过量化方法, 取量化步长为 2^{-n}, 通过公式 6 计算 n-bit 精确的香农熵[22], 其中 p_i 是 X 的取值在第 i 个量化区间的概率. 香农熵衡量随机变量的无序程度.

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{6}$$

CPU 资源是计算系统中最重要的资源之一, 也需要有类似的值来衡量 CPU 管理和使用的无序程度, 本文称之为 CPU 熵.

本文定义 CPU 熵为将 CPU 指标视为随机变量后计算得到的香农熵. 对于 CPU 指标 I, 通过量化方法, 取量化步长为 Δ, I 在第 i 区间出现的概率为 $p_i \in [0,1]$. 本文以 0.01 的步长量化 CPU 指标 I, 例如 CPU 有效率将被划分为 100 份.

定义 5. 两个 CPU 比率的香农熵（简称熵）. 从公式（6）, 可定义两个熵, 分别刻画使用率 U、有效率 V 的无序程度, 称为使用熵 H_U、有效熵 H_V, 统称为 CPU 熵.

$$H_U = -\sum_{i=1}^{N} p_{ui} \log_2 p_{ui} \tag{7}$$

$$H_V = -\sum_{i=1}^{N} p_{vi} \log_2 p_{vi} \tag{8}$$

其中 p_{ui} 和 p_{vi} 分别为指标 U 和 V 的取值出现在第 i 个量化区间的概率.

3.4 CPU资源的两个正则化熵

依据香农熵定义的 CPU 熵，在一定程度上描述了系统 CPU 使用和管理的无序程度，并在后续实验中从相关性上支持了“低熵假设”，但是未能准确地刻画系统是否高效地使用和管理 CPU 资源. 例如 K8s-1C2C（请求 1 核、上限 2 核）在高负载下 CPU 有效率一直接近于 0，通过香农熵计算的 CPU 有效熵也接近 0，但是显然 CPU 使用并不高效. K8s-4C8C（请求 4 核、上限 8 核）在峰值 90，加速度 0.2 的渐变负载下，其 CPU 有效率的概率密度函数在 0%、21%、45%形成 3 个峰. 如果这三个峰发生聚集或分离，香农熵的值并不会发生变化，但是 CPU 管理和使用的无序程度是不一样的. 所以通过香农熵还不足以衡量系统使用 CPU 的无序程度，需要将其正则化来矫正上述问题. 本文通过先验知识和正则化熵三条件来辅助定义正则化的 CPU 熵.

本文采用贝叶斯学派关于正则化的观点[23]，即通过加入先验知识来正则化基于香农熵定义的 CPU 熵，以更精确刻画系统对 CPU 管理使用的无序程度. 本文加入两条先验知识：其一是 CPU 使用率或有效率越高，其对应的正则化熵越低；其二是 CPU 使用率或有效率越聚集，其对应的正则化熵越低.

根据上述先验知识，给出单个 CPU 指标熵的正则化方法. 对于 CPU 指标 I，通过量化方法，取量化步长为 $\Delta=0.01$，第 i 区间的中值为 I_i，I 在第 i 区间出现的概率为 $p_i\in[0,1]$，I_i 与该指标其他取值的聚集程度为 a_i，则定义该指标的正则化熵 R_I 为

$$R_I=-\sum_{i=1}^{N}q_i\log_2(q_i),\quad q_i=p_i\times f_I(I_i,a_i) \tag{9}$$

R_I 描述指标 I 的无序程度，R_I 越大则指标 I 无序性越高. 给出上述定义的合理性在于（1）$-\sum_{i=1}^{N}q_i\log_2(q_i)$ 在形式上与香农熵一致；（2）上述正则化是根据先验知识对概率的加权修正，使得新定义的正则化熵在“概率”维度具备与香农熵相似性质，即概率分布越分散，R_I 的值越大. $f_I(I_i,a_i)$ 的选取应当满足先验知识 1 和 2. 为了保证正则化后的熵大于等于 0，要求 $f_I(I_i,a_i)$ 的取值在 $[0,1]$ 区间.

定义 6. 两个正则化的 CPU 熵. 根据公式（9），分别对使用熵 H_U 和有效熵 H_V 正则化，得到正则化的使用熵 R_U、正则化的有效熵 R_V. 应当选取合适的聚集程度归一化指标 $a_i\in[0,1]$，使得 a_i 越大可得 I_i 与该指标的其他取值的聚集性越低. R_U 和 R_V 的定义如下：

$$R_U=-\sum_{i=1}^{N}q_i\log_2 q_i,q_i=p_i\times f_U(U_i,a_i) \tag{10}$$

$$R_V=-\sum_{i=1}^{N}q_i\log_2 q_i,q_i=p_i\times f_V(V_i,a_i) \tag{11}$$

其中，选取的 f_U、f_V 应当使得对应的 $-q_i\log_2 q_i$ 满足以下三个条件(令 $I_i=U_i$或$I_i=V_i$)，以满足先验知识，称为正则化 CPU 熵三条件. 其中条件 1、2 分别对应先验知识 1、2，条件 3 给出了正则化熵的下界以及达到下界的条件.

条件 1. 对于任意 p_i 和 a_i，函数 $-q_i\log_2 q_i$ 关于变量 I_i 在 $[0,1]$ 区间单调递减. 含义为对于相同的概率分布，指标 I 越接近 1，熵越低.

条件 2. 对于任意 p_i 和 I_i，函数 $-q_i\log_2 q_i$ 关于变量 a_i 在 $[0,1]$ 区间单调递增. 含义为指标 I 越聚集，该指标对应的熵越低.

条件 3. 对于任意 p_i，当 $I_i=1,a_i=0$ 时，函数 $-q_i\log_2 q_i$ 的值为 0. 含义为当指标 I_i 为 1 且最聚集时，对应的熵为 0.

为简单起见，本文仅考虑 f_I 是连续可导的情况. 则条件 1 要求在保持 p_i 和 a_i 不变的情况下，函数 $-q_i\log_2 q_i$ 对 I_i 的导数恒小于 0. 条件 2 要求在保持 p_i 和 I_i 不变的情况下，函数 $-q_i\log_2 q_i$ 对 a_i 的导数恒大于 0. 条件 3 要求可得出 $f_I(1,0)=0$.

先考虑条件 1，$-q_i\log_2 q_i$ 对 I_i 的导数为

$$-p_i\times\left(\log_2(p_i\times f_I(I_i,a_i))+\frac{1}{\ln 2}\right)\times\frac{\partial f_I(I_i,a_i)}{\partial I_i}.$$

由于给定任意 p_i，上述导数恒小于 0，即取 p_i 无限接近 0 也成立，说明 $\frac{\partial f_I(I_i,a_i)}{\partial I_i}$ 的值对于任意 a_i,I_i 都应小于 0. 因此可进一步得出 $-\left(\log_2(p_i\times f_I(I_i,a_i))+\frac{1}{\ln 2}\right)$ 应当大于 0，可以导出选取的 f_I 应使得 $p_i\times f_I(I_i,a_i)<\frac{1}{e}$ 恒成立. p_i 的最大值为 1，因此对于任意 I_i,a_i 的 $f_I(I_i,a_i)<\frac{1}{e}$.

综上，对于任意 a_i,I_i，$\frac{\partial f_I(I_i,a_i)}{\partial I_i}$ 恒小于 0；对于任意 I_i,a_i，$f_I(I_i,a_i)<\frac{1}{e}$.

由于对称性，同理可得，对于任意 a_i,I_i，$\frac{\partial f_I(I_i,a_i)}{\partial a_i}$ 恒大于 0.

存在多种函数 f_I 满足正则化 CPU 熵三条件，本文考虑了一次、二次多项式和指数形式，对应形式分别为 $\frac{(1+a_i-I_i)}{8}$，$\frac{(1+a_i^2-I_i^2)}{8}$ 和 $\frac{(2+2^{a_i}-2^{I_i})}{16}$，分母为满足条件的最小 2 的幂次. 图 2 给出了三种形式在概率 $p=0.5$ 和聚集程度 $a=0.5$ 时，函数 $-q\log_2 q$ 随指标 I 的变化图. 因一次多项式的计算最简单，故我们初步选取 $f_I(I_i,a_i)=\frac{(1+a_i-I_i)}{8}$. 本文采用加权距离来衡量上述指标的聚集程度：$a_i=\sum_{n=1}^{N}p_n\times|I_i-I_n|$，(令 $I_i=U_i$或$I_i=V_i$). 图 3 给出了上述函数 $-q\log_2 q$ 在概率 $p=0.75$ 和选取的 f_I 的情况下随聚集程度 a 和指标 I 的变化曲面图.

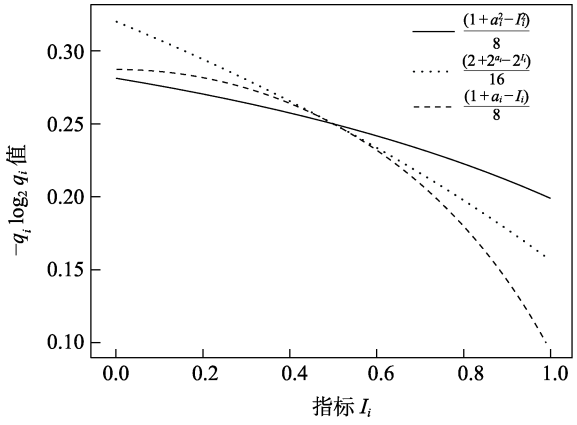


图 2 正则化熵三种形式的对比

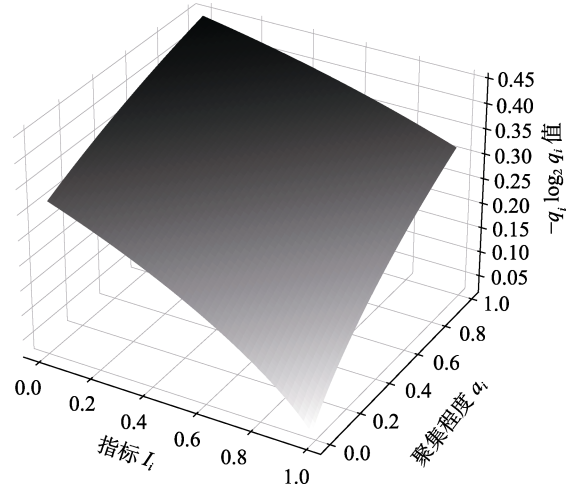


图 3 一次多项式的正则化熵三维曲面图

4 实验系统设计

如图 4 所示，整个实验系统由负载发生器、信息高铁原型、K8s 系统、日志分析器以及相关硬件设备组成. 负载发生器会生成基本一致且公平的计算负载给信息高铁原型和 K8s 系统. 两个系统使用相同的计算、内存和网络资源. 不同之处是，信息高铁原型使用全球导航卫星系统（Global Navigation Satellite System，GNSS）和精确时间协议（Precision Time Protocol，PTP）来保证分布在不同地理位置的算力资源的时钟一致性，而 K8s 系统则使用网络时间协议（Network Time Protocol，NTP）进行授时. 日志分析服务根据运行日志来计算两个系统的良率、通量和 CPU 相关指标等. 日志的时间戳采用基于 PTP 授时机制提供的时间. 为避免多个系统运行时的互相干扰，每次实验仅启动信息高铁原型或 K8s 系统中的一个.

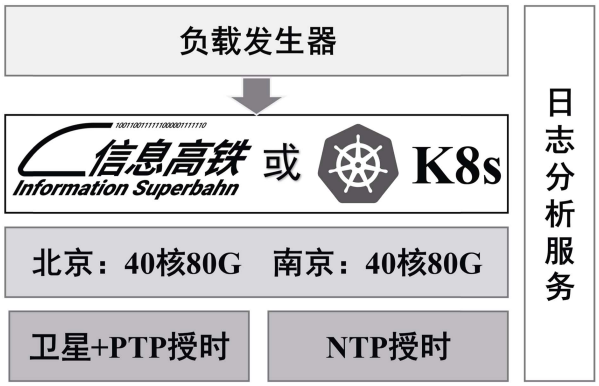


图 4 实验系统总览

下面详细阐述我们根据信息高铁设计和建设的指导思想做出的两个重要的原型设计决定：

决定 1：基于跨域共享资源的快速缩扩容. 所有计算资源被组织成一个跨域分布式资源池，任一计算任务都可以在任一资源上执行. 从逻辑上看，用户通过向一个大队列提交计算任务，共享使用分布式资源池，避免了简单的切片式分配资源的“线路交换”陷阱. 每一个任务在被调度时，均会决定是否进行资源扩容且扩容时间可低至 10 毫秒级别，相比于 K8s 系统最高 1 秒检测一次是否需要扩容更加精细和准确. 上述决定可以支持原生的跨域任务调度并且降低应扩容不及时导致的任务拥塞，进而提高通量和良率.

决定 2：原生提供 DIP 支持. 每一个计算任务都具有全局唯一的标签以支持“区分”能力. 该标签在任务发起时就已经确定，并用于任务的调度和执行. 原型系统按照计算任务的需求来分配 CPU 资源，任务在执行时独占使用 CPU 资源以支持“隔离”能力. 原型系统提供最高、次高和最低三种优先级以支持“优先化”能力；同时对于相同优先级任务，

要求提供任务的估计执行时间和截止时间（依赖于全局一致时钟），并基于最短剩余时间的优先调度算法进行调度. 以上设计基于 DIP 猜想，通过支持区分、隔离和优先化来提供低熵有序的资源共享，降低系统各种无序对用户体验的影响.

其他决定包括基于最小化网络开销和全局状态的跨域分散式调度、基于高频上报的资源状态的负反馈负载均衡机制、相同优先级任务的执行顺序动态调整机制等.

4.1　实验角色与任务类型

实验包含物端、开发团队、高级用户、独占用户四类角色. 开发团队发送实验命令和接收任务结果，其余三者接收开发团队的消息，并根据消息内容提交计算任务. 因此，系统的任务分为物端任务、高级用户任务和独占用户任务三类.

实验要求独占用户具有最高优先级，高级用户的任务具有次高优先级，物端任务优先级最低. 独占用户任务执行时不允许其他用户的任务执行. 当高级用户任务和物端的任务同时到达计算系统时，应当优先执行高优先级用户的任务. 虽然 K8s 系统可以提供容器级别的优先级调度，但是无法针对单个任务请求进行优先调度.

实验的每次测试分为测试任务阶段和红包任务阶段. 负载发生器在测试任务阶段向系统发送 N 个测试任务，所有测试任务执行完毕后生成“执行凭证”（包含系统良率、通量等），主要目的是验证信息高铁原型的良率和通量比 K8s 系统有数量级提升；红包任务阶段是收到“执行凭证”后，不同角色的用户发送由红包任务构成的负载，用于验证信息高铁原型提供的优先级保障能力. 具体流程将在 4.3 节详细说明.

测试任务和红包任务均为手写数字识别任务，识别算法采用全连接神经网络模型. 物端、高级用户和独占用户均向信息高铁原型或 K8s 实验系统提交 28 像素 × 28 像素的 PNG 图片进行手写数字识别. 手写数字识别结果将作为红包金额发送给开发团队. 单独执行 1 个手写数字识别任务，南京 1-5 号服务器、北京 1 号服务器和北京 2 号服务器分别平均需要约 8.17 ms、6 ms 和 4.9 ms.

4.2　实验环境配置

实验系统包括 7 台 x86 服务器，10 台 NVIDIA Jetson 物端板卡，54 台虚拟物端，2 台 PTP 交换机和 2 台卫星授时模组.

上述服务器中的 2 台部署在北京，5 台部署在南京. 它们分别被命名为北京 1-2 号服务器和南京 1-5 号服务器. 北京和南京的服务器集群之间采用公网连接，由科技网和联通提供公网服务. 服务器的具体硬件配置和网络参数参见表 2 和表 3. 64 台物端中，南京部署 10 台真实物端和 14 台虚拟物端，北京部署 40 台虚拟物端.

表 2　服务器配置

服务器名	CPU	内存
北京 1 号服务器	40 核 Intel Xeon E5-2640	128GB DDR4 2400
北京 2 号服务器	160 核 Intel Xeon 6138	128GB DDR4 2666
南京 1-5 号服务器	96 核 Intel Xeon 8163	256GB DDR4 2666

表 3　网络参数

网络	带宽	单向延时
南京到北京	8.45 Mbps	13.7 ms
北京到南京	41.5 Mbps	13.7 ms
集群内部	941 Mbps	0.11 ms
南京：服务器到物端	941 Mbps	0.19 ms

每台服务器均配置 PTP 和 NTP 两个授时模块. NTP 授时模块从网络 NTP 授时服务器获取时间. PTP 授时模块使用 GNSS 模组和 PTP 协议同步得到的全局一致时钟. 在 PTP 授时模式下，同机房内的设备时钟偏差为 10 微秒级，北京南京两地机房间的时钟偏差不超过 5 毫秒.

为了确保对比实验的公平性，我们为信息高铁原型和 K8s 系统配置了相同的计算、存储资源和网络环境. 信息高铁原型和 K8s 系统分别部署在北京和南京各两台服务器上，每台服务器限制使用 20 核 CPU 和 40GB 内存用于执行任务.

我们选择了 2 种典型的 K8s 容器配置作为对比系统. 配置分别为 K8s-1C2C：请求 1 核+上限 2 核+60%扩容；K8s-4C8C：请求 4 核+上限 8 核+10%扩容. 扩容策略为每秒检查是否可以扩容，单个 K8s 集群最高可扩容 1000 个容器. 扩容检测频率为 K8s 可提供的最高频率，即 1 秒一次.

本实验使用的 K8s 版本号为 1.23，发布日期为 2021 年 12 月. K8s 系统初始在南京和北京的 K8s 集群各部署一个容器. 物端启动时会发送一个预热任务，使得信息高铁原型和 K8s 系统中都在本地缓存有执行环境.

根据调研发现，相关工作在进行真实的跨数据中心实验时，采用的总 CPU 数为 100 核左右[24-25]，与本文的系统资源总量相当. 同时本文主要目的是验证信息高铁的高通量潜力. 所以上述资源可以支撑本文的研究和实验. 我们将在未来工作中使用更多资源构建信息高铁原型系统.

4.3 实验流程及测试负载设计

本节介绍负载发生器生成负载的具体过程. 信息高铁原型和 K8s 系统的负载任务相同，生成方式基本一致. 接下来主要从信息高铁原型的角度介绍测试负载生成方法，并给出 K8s 实验系统的负载生成与上述方法的不同之处.

信息高铁原型负载生成流程如图 5 所示. 实验细分为四个步骤，其中步骤 1 和 2 是执行测试任务阶段，步骤 3 和 4 是执行红包任务阶段：

（1）开发团队发送“测试通知”，红包应用服务将“测试通知”广播给 64 个物端.

（2）64 个物端启动负载生成服务，生成大量的测试任务并发送给信息高铁原型. 日志分析服务会生成“执行凭证”，包含了系统良率、通量、CPU 使用指标和熵等. 在多轮实验中，物端会分别采用突发负载和渐变负载两种负载特征. 同一轮实验中，所有物端发送任务的特征一致.

（3）开发团队向高级用户、独占用户和物端分别发送附带“执行凭证”的信息高铁“开通消息”.

（4）独占用户、高级用户和物端发送红包任务给红包应用服务，其将任务转发给信息高铁原型，执行结果会发送给开发团队. 上述三类角色根据“开通消息”携带的发送时刻 T，决定各自的任务发送时刻. 其中独占用户在其本地时间为 T+25ms 时刻发送 1 个红包任务，高级用户和物端在其本地时间为 T+55ms 时分别发送 1 个和 5 个红包任务. 三个角色的红包任务执行结果将作为红包的数额，显示在开发团队前端页面. 实验将红包到达开发团队前端界面的次序作为系统确保任务优先级的凭证.

K8s 系统负载生成流程如图 6 所示. 生成过程与信息高铁原型的不同处均用灰色线圈出，包括以下三点：

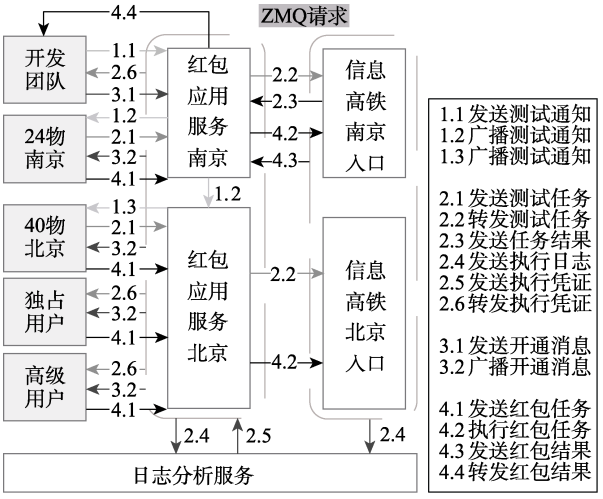


图 5 信息高铁原型负载生成工作流

（1）向 K8s 系统发送任务时，使用互联网中常用的 HTTP 请求；而向信息高铁原型发送任务时，使用基于 ZMQ[①]的信息高铁原型任务协议.

（2）K8s 系统直接使用 HTTP 的响应来发送手写数字识别结果；信息高铁原型要求任务需要携带结果接收地址，并通过基于 ZMQ 的协议将结果发送给接收方.

（3）K8s 系统的请求仅包含一张手写数字图片信息；信息高铁原型的任务请求除了上述图片外，还可能包含任务的元信息和可执行代码.

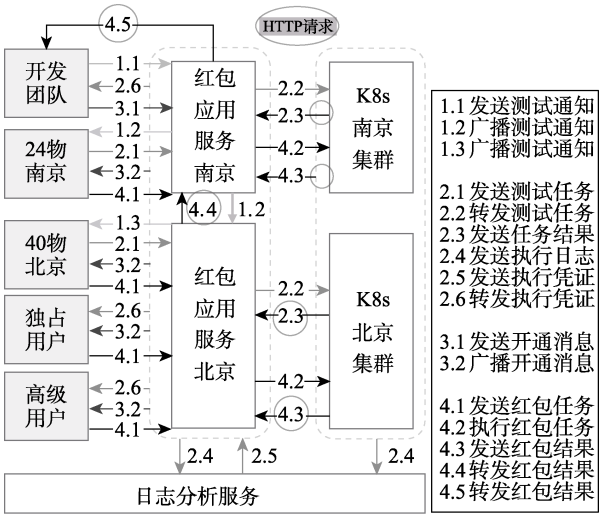


图 6 K8s 负载生成工作流

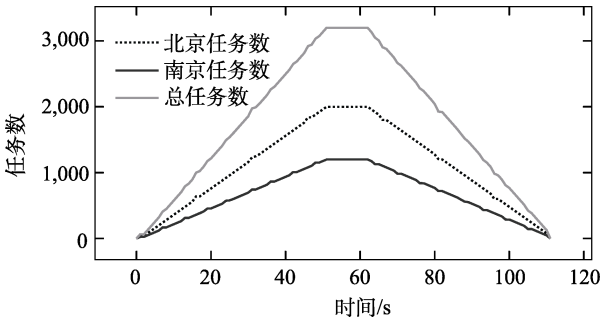


图 7 渐变负载特征（任务峰值 50，加速度 1）

为了模拟智能万物互联时代的负载变化特征，实验设计了突发、渐变两种特征的测试负载.

（1）突发负载是云计算系统的典型负载，例如电商的购物节秒杀活动会产生的大量商品浏览、交易等计算任务. 大量用户会在某一时刻同时向系统发起海量并发请求，并会持续一段时间.

（2）渐变负载的特征是请求持续上升到峰值或者从峰值逐渐下降. 例如，热搜请求会在几个小时

① ZeroMQ, https://zeromq.org/ 2022,7,25

内逐渐增加到达顶点，之后又逐渐下降. 这类负载可以评价系统应对负载变化能力.

另外，我们使用不同优先级和保质要求的负载来综合评价信息高铁原型和 K8s 系统的保质和优先化能力. 根据物体实时检测的保质要求[26]，分别设置 100 ms、50 ms 和 20 ms 共 3 档保质时间要求. 北京与南京红包应用服务发起的 100 ms、50 ms 和 20 ms 保质要求的任务数比例分别为 1:1:0 和 2:1:1. 相同保质要求的任务在时间上均匀分布.

实验采用的突发负载为：多个物端在 40 秒内以恒定速率发送若干数量的任务. 本次实验包含多个独立测试，每次测试发送的任务总量从 100 到 365696 不等，信息高铁原型额外测试到 853376 个任务. 以 102400 个任务为例，南京和北京的红包应用服务每隔 25 ms 分别提交 24、40 个任务.

渐变负载是连续梯形特征的负载. 该负载由任务发送峰值速率（下称任务峰值）p 和任务发送加速度（下称加速度）a 决定. 任务峰值为一个物端每秒发送任务速率的峰值. 加速度为一个物端每秒增加的提交任务数. 该负载使用 64 个物端，每个物端以 a 的加速度发送任务，直到发送速率达到 p，并持续 10 秒，然后以–a 加速度发送任务，直到发送速率降为 0. 该实验同样包含多个独立测试，分别是加速度为 0.2、1、10，任务峰值为 10 到 170 的情况. 图 7 给出了实际测试时任务峰值为 50，加速度为 1 的负载变化特征图，64 个物端共发送 195200 个任务.

上述每一个测试即构成一个负载. 所有的测试构成全部负载，并被划分为低负载和高负载两部分. 我们将预估峰值 CPU 需求为 25%总 CPU 资源量（简称 25%阈值）作为判别高低负载的分界线. 结合实验中负载任务在各型号执行机上的平均执行时长，我们计算得出：40 秒内发送总任务数高于 123080 的突发负载为高负载；任务峰值高于 48 的渐变负载为高负载. 由于信息高铁是一个面向高通量计算场景的基础设施，我们重点考虑和研究高负载下信息高铁的良率、通量以及熵和两者的关系.

我们通过相关工作和论证分析，说明 25%阈值的合理性. 在商业化运行的云计算集群中，为了保证时延敏感型(LC)任务的要求，加之内存读写、网络传输等部分的限制，在季节性购物节期间，LC 实例的 CPU 平均利用率约为 25%[27]. 由于在该 CPU 利用率情况下，可以保障用户体验，故我们将 25%阈值作为判别高负载的下界. 同时云提供商会设置 LC 实例的过载条件（例如 50% CPU 利用率）[28]. 在附录 A 中，讨论了选取不同阈值的情况. 结果表明，在 25%~50%之间选取不同的阈值，不会影响本文对三个假设的验证结果. 因此，针对本文的具体情况，我们认为选取 25%作为阈值是合理的. 但是，该定义并不在普遍的情况下适用，在其他场景中通常还需要综合考虑 IO、内存使用、网络带宽等. 高低负载的量化定义是一个值得研究的问题.

以下通过实验的工作量和相关工作说明本文的实验负载可以支撑本文研究问题. 我们考虑了服务计算常见的突发和渐变两种负载特征（共执行了 40 多组实验，数百小时测试时长），并采用人工智能领域中具有代表性的手写数字识别任务作为真实计算负载. 据调研发现，著名的大数据处理框架 Spark[29] 首次在论文中展示其性能时，使用的负载类型数量与本文相当. 同时本文的主要目的是回答：信息高铁是否具备数量级提升通量的潜力？因此，采用上述负载可以验证信息高铁的潜力.

4.4　信息高铁原型系统实现

信息高铁的核心思想是将广域分布的计算资源看成一个整体，通过有序的、精细化的计算时空[4]管理，统一处理用户提交的所有任务. 从逻辑上看，所有提交的任务都被存放在一个全局任务大队列中，信息高铁原型根据任务优先级、资源状态和任务需求等决定任务执行顺序，使无序任务请求转变为有序的任务执行.

信息高铁原型以更小的任务粒度进行资源分配，实现不同负载的任务时空复用同一资源，从而实现资源的高效利用. K8s 系统则是以容器（服务实例）的粒度进行资源分配，为保证服务质量，即使没有任务请求也会被分配资源，造成资源的“占而不用”问题.

如图 8 所示，信息高铁原型采用基于全局一致时钟的四级调度架构，分别是信息高铁入口、全局状态存储、信息高铁站点（简称站点）和执行机. 所有的信息高铁入口构成了全局唯一任务大队列的入口. 全局状态存储为信息高铁入口提供全局信息，实现基于全局资源信息的任务调度. 站点负责管理单个集群的任务调度，允许不同运营商使用不同的运营策略. 执行机负责管理单台机器的任务执行，是最小管理单元.

具体而言，用户提交任务到返回结果的处理流程如下：

（1）用户向信息高铁入口提交任务. 该入口根据全局状态存储提供的站点状态、任务需求、优先级等信息调度任务到合适的站点.

（2）站点每 5 毫秒执行一次调度，根据执行机

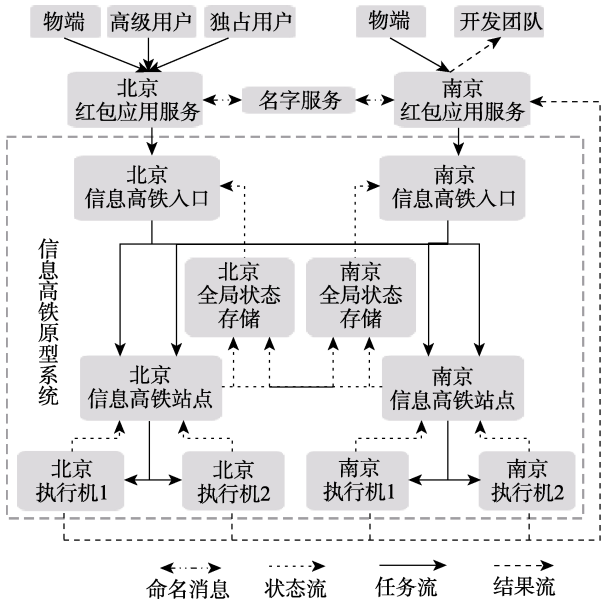


图 8 信息高铁原型系统架构

资源的当前状态和未来预估状态、任务优先级、保质要求等信息对任务进行编排，然后发送给相应的执行机. 本实验采用的编排策略是在保质的前提下最大化利用系统的资源时空[4].

（3）执行机创建执行环境，即执行任务代码的运行实例（worker）. 执行机先根据任务的元数据（CPU 核数、预估执行时长等）分配资源，而后根据任务参数和任务可执行代码在执行环境中启动任务. 任务运行结束后，系统回收资源并依据任务元数据提供的地址将结果发送给接收方.

执行机每 5 毫秒将未来 5 毫秒的可用资源上报给对应的站点；首次向信息高铁原型提交任务时，会附带该任务运行需要的所有依赖（包括库、配置文件等），即执行环境. 信息高铁原型支持运行依赖和任务可执行代码的缓存，同类任务的可执行代码和依赖无需重复提交.

信息高铁原型采用 C 语言实现，源代码约 6400 行（SLOC）. 信息高铁原型各个子系统通过基于 CZMQ①库实现的消息队列进行通信. 独占用户、高级用户和开发团队的前端采用 QT②框架实现，源代码约 1580 SLOC，红包应用服务采用 Go 语言实现，源代码约 3600 SLOC. 实验代码、数据和可执行文件已经在 github 上发布③. 横跨北京-南京的广域分布式实验环境也会在未来开放，以便为学术界提供实验和验证环境.

5 实验结果分析

5.1 节汇总了实验结果. 5.2~5.4 节分析实验结果，讨论实验数据如何验证了高通量假设、低熵假设、优先化假设. 5.5 节讨论了实验结果显示的信息高铁原型呈现出的其他特征.

5.1 测量结果

我们在南京–北京实验环境上总共做了 300 余次测试，每次费时 100 秒至 3 小时不等. 图 9、图 10、图 11 展示了测试结果数据. 图中的一个点为实验的一个样本.

图 9(a)展示了突发负载下信息高铁原型和 2 种配置的 K8s 系统的良率和通量变化情况. 图 9(b)、图 9(c)、图 9(d)分别展示了渐变负载下，加速度为 0.2、1、10 时，信息高铁原型和 2 种配置的 K8s 系统在任务峰值为 10~170 时的良率和通量变化情况.

图 10(a)展示了突发负载下信息高铁原型和 2 种配置的 K8s 系统的 CPU 使用熵和 CPU 有效熵的变化情况. 图 11(a)为对应的正则化后的 CPU 使用熵和 CPU 有效熵的变化情况.

图 10(b)、图 10(c)、图 10(d)分别展示了渐变负载下，加速度为 0.2、1、10 时，信息高铁原型和 2 种配置的 K8s 系统在任务峰值为 10~170 时 CPU 使用熵和 CPU 有效熵变化情况. 图 11(b)、图 11(c)、图 11(d)为对应的正则化后的 CPU 使用熵和 CPU 有效熵的变化情况.

5.2 高通量假设验证

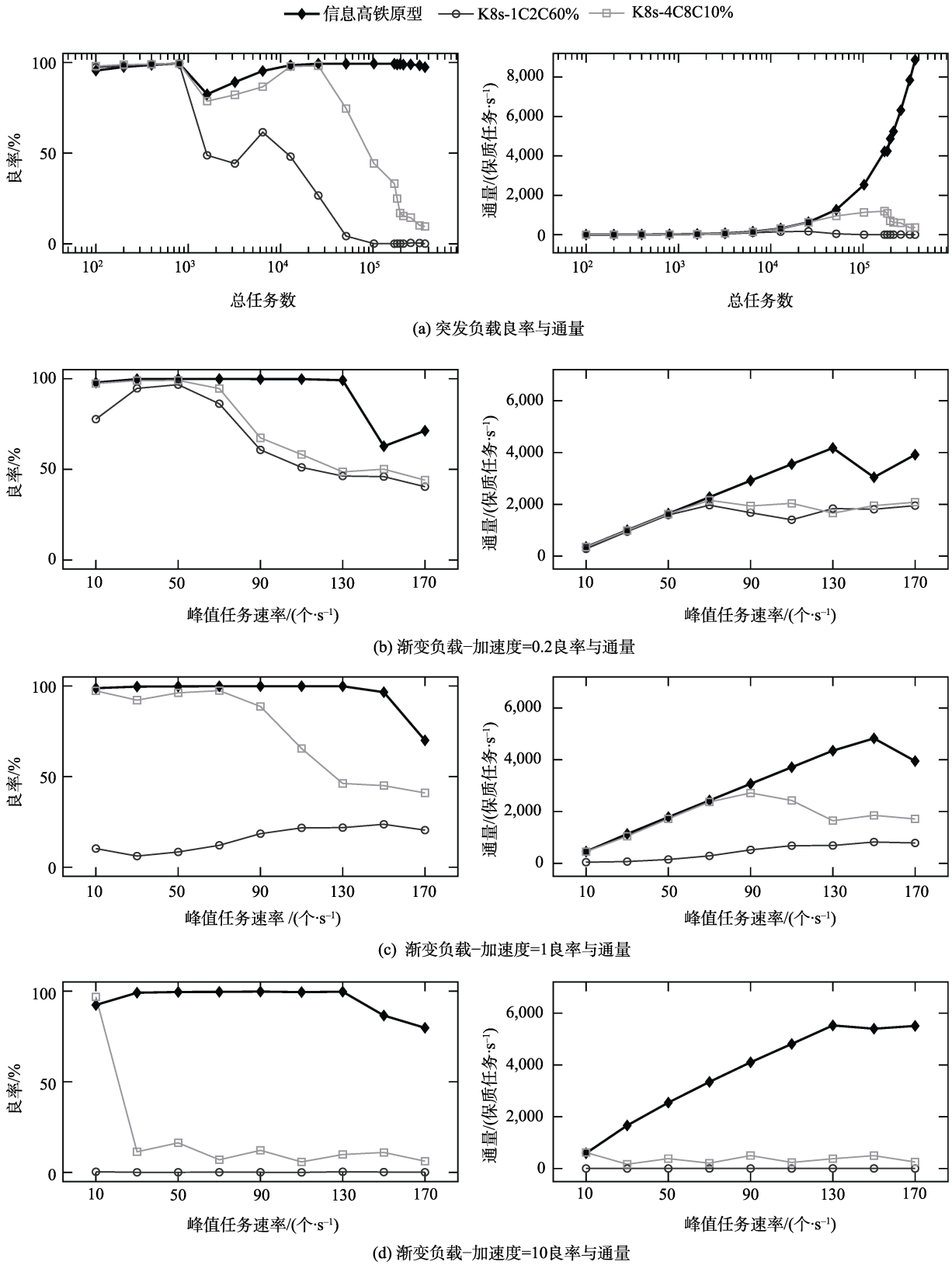
300 多次测试产生的实验数据显示，信息高铁原型在 92.0%的样本中实现了比 K8s 系统更高的良率，分别在 55.5%、25.5%、18.9%、12.2%的样本中良率提升超过 1 倍、10 倍、100 倍、1000 倍. 信息高铁原型在 92.0%的样本中实现了比 K8s 系统更高的通量，分别在 58.9%、28.9%、18.9%、14.4%的样本中通量提升 1 倍、10 倍、100 倍、1000 倍. 仅在 8.0%的样本中，信息高铁原型的良率或通量低于 K8s 系统，最多下降 4.6%，均为低负载的情况且信息高铁原型良率均高于 98%.

在高负载情况下，信息高铁原型的良率和通量提升尤其明显. 例如，在突发高负载情况下，当任务数为 256000 时，信息高铁原型的良率是 K8s 系统的 6.8~210 倍，通量是 10~631 倍. 在渐变负载情况下，当加速度为 10，任务峰值为 130 时，信息高铁原型的良率是 K8s 系统的 10.9~278 倍，通量是

① CZMQ, http://czmq.zeromq.org/ 2022,7,25

② QT, https://www.qt.io/ 2022,7,25

③ red-packet-exp, https://github.com/yzs15/red-packet-exp 2023, 3, 22



(a) 突发负载良率与通量

(b) 渐变负载−加速度=0.2良率与通量

(c) 渐变负载−加速度=1良率与通量

(d) 渐变负载−加速度=10良率与通量

图 9　两种测试负载下系统良率与通量

14.6~705 倍.

5.3　低熵假设验证

低熵假设的验证分为两个方面：其一为探讨 CPU 熵与良率、单核通量的关系，需要验证信息高铁原型的良率和单核通量均与 CPU 熵负相关；其二是验证信息高铁原型的 CPU 熵降至 K8s 系统的 50%

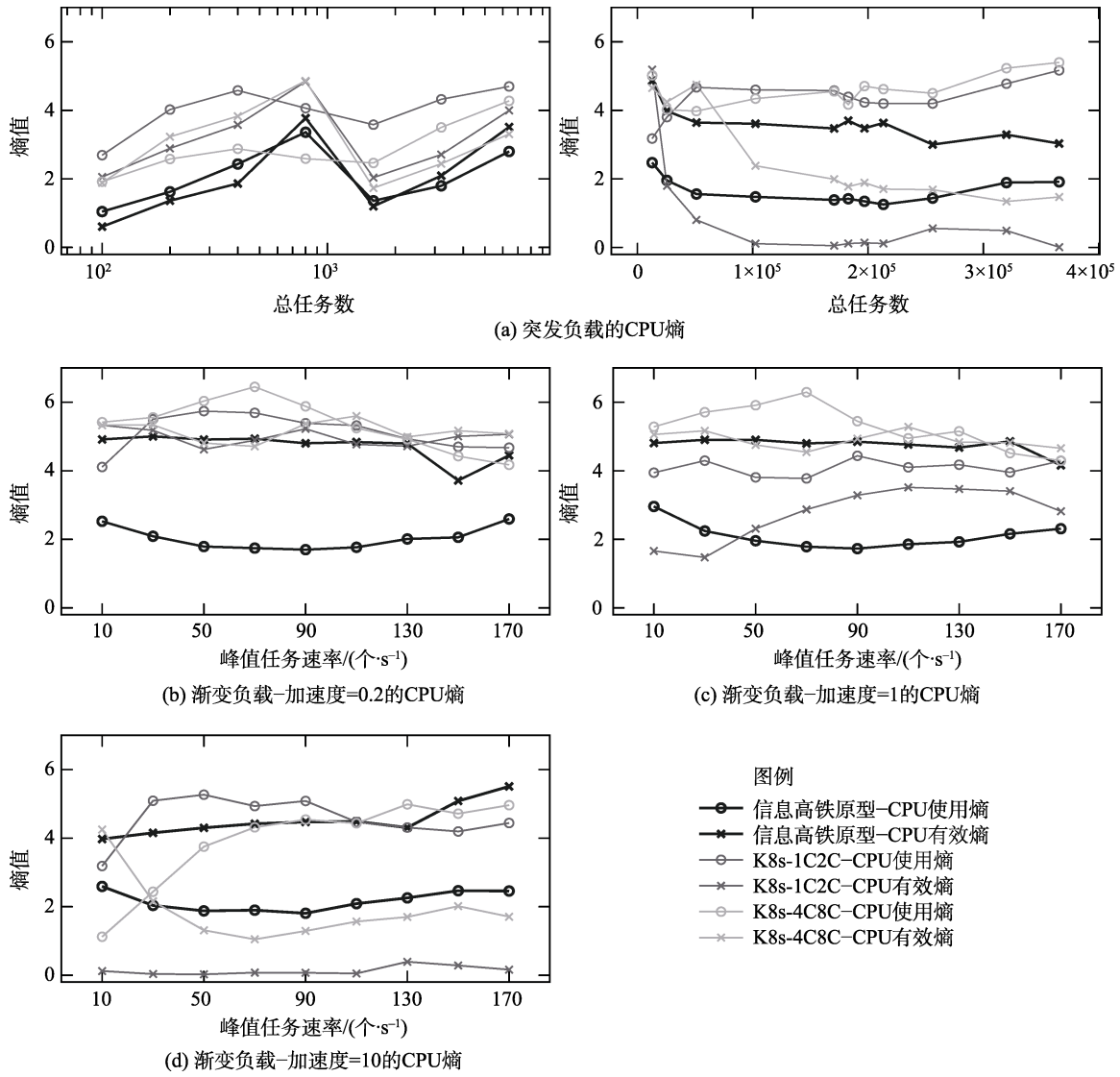


图 10 两种测试负载的 CPU 熵

表 4 计算 Spearman 系数的测试点数量

负载类型	1C2C	4C8C	信息高铁原型
低负载	34	34	34
高负载	56	56	76
全部负载	90	90	110

以下.

5.3.1 良率和通量与 CPU 熵的关系

我们计算了单核通量和良率与 CPU 熵的 Spearman 等级相关系数[30]. 结果表明，参照文献[31]中对相关系数的解释，信息高铁原型的良率和单核通量分别与 CPU 熵呈现显著的强负相关和中等强度负相关. 换言之，CPU 熵越低，单核通量与良率越高. 这从相关性角度支持了低熵假设. 但是相关性分析并不能得出单核通量和良率与 CPU 熵是否存在因果关系的结论. 我们将上述三者的因果关系研究留给未来工作.

从实验数据可计算出良率和单核通量与 CPU 熵的 Spearman 相关系数. 本文使用高负载的数据、低负载数据和全部数据分别计算之，表 4 给出了上述三种情况的测试点数. 高负载和全部负载的结果见表5~表8. 以CPU使用熵和良率为例说明Spearman相关系数的计算方法：总共 300 多次测试，每次测试均可以得到 CPU 使用熵和良率的一组数据，每组数据构成一个点；分别使用所有点、高负载对应的点、低负载对应的点来计算全部负载、高负载、低负载下 CPU 使用熵和良率的 Spearman 相关系数. 结果表明在高负载下，信息高铁原型的良率和单核通量分别与两种 CPU 熵呈现显著的强和中等程度负相关（p<0.001）. 这从相关性角度揭示了低熵化方法具备提升系统通量和良率的潜力. 在全部负载

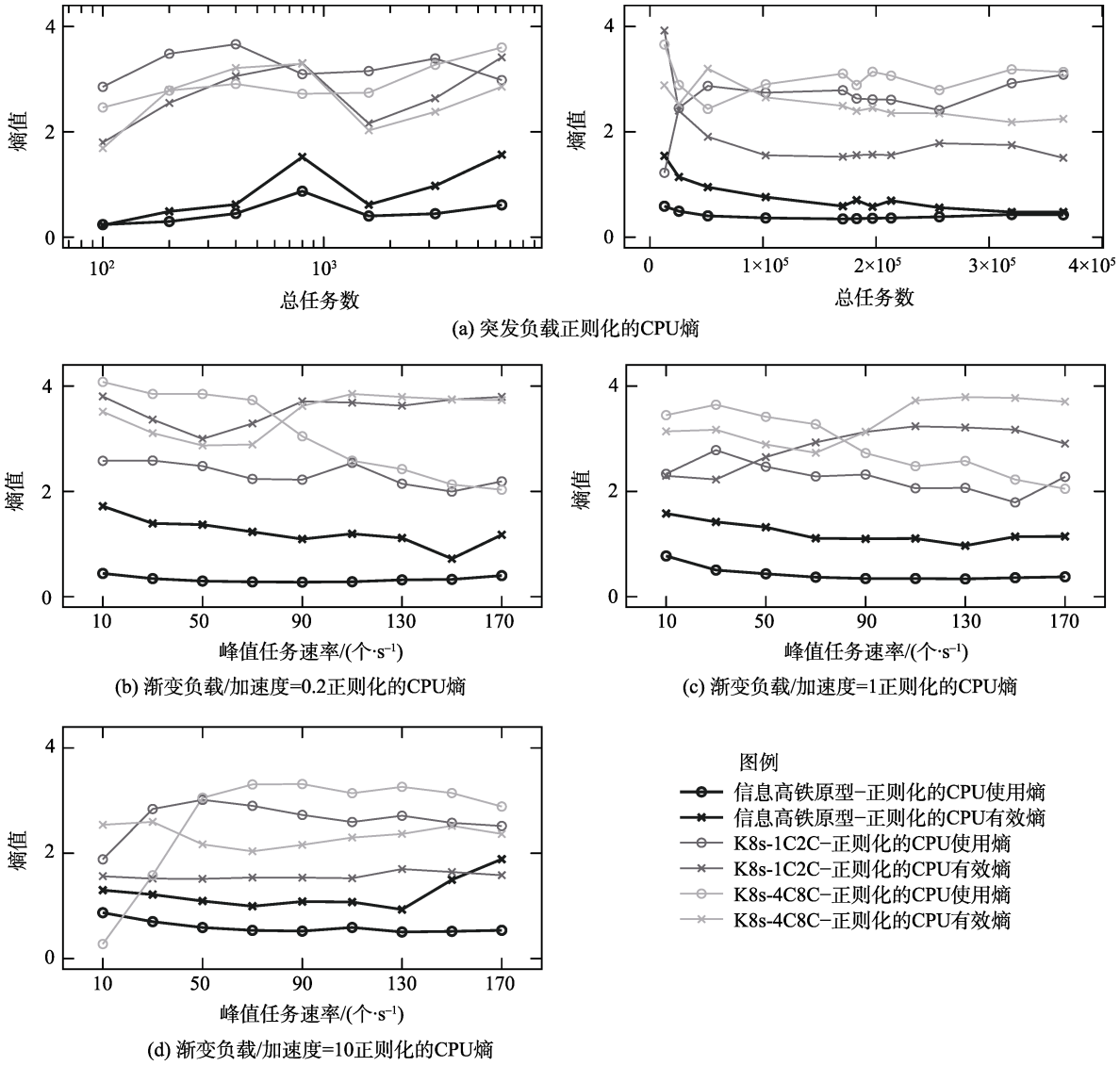


图 11　两种测试负载的正则化 CPU 熵

下，信息高铁原型的 CPU 熵，除了单核通量与有效熵无明显相关性外，其余均为中等强度负相关. 在低负载时，则无明显相关性. 在低负载下，除 4 个样本点外，系统良率均接近 1（均值=0.9858，标准差=0.0118），因此推测相关性分析存在较大误差. 需要说明的是，低负载下的相关性并非本文关注重点，故不做详细探讨. 由于低负载下的无相关性，且高低负载共同构成全部负载，导致全部负载下的相关性会低于高负载下的相关性. 信息高铁作为高通量算力基础设施，应当主要关注高负载下的表现.

5.3.2　正则化的 CPU 熵

在突发负载和渐变负载条件下，信息高铁原型正则化后的 CPU 熵降至 K8s 系统的 50%以下，支持了低熵假设. 在比较正则化的有效熵时，K8s 与信息高铁原型均采用估计的有效核数.

图 11(a)展示了突发负载时两个正则化的 CPU 熵的变化情况. 实验结果显示信息高铁原型在全部样本中具有更低的正则化的 CPU 使用熵和有效熵. 正则化的 CPU 使用熵、有效熵分别在 100%、88.89%的样本中降低 50%，在 91.7%、44.4%的样本中降低 75%.

图 11(b)、图 11(c)、图 11(d)展示了渐变负载时两个正则化的 CPU 熵的变化情况. 结果显示信息高

表 5　良率与 CPU 熵的关系（高负载，p<0.001）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.30(p=0.026)	0.54	**–0.71**
有效熵	0.94	0.81	**–0.86**
正则化的使用熵	–0.65	0.04(p=0.78)	**–0.50**
正则化的有效熵	0.91	0.72	**–0.92**

表 6 单核通量与 CPU 熵的关系（高负载，p<0.001）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.33(p=0.014)	0.37(p=0.005)	**–0.58**
有效熵	0.92	0.58	**–0.60**
正则化的使用熵	–0.61	0.11(p=0.43)	**–0.38**
正则化的有效熵	0.88	0.48	**–0.55**

表 7 良率与 CPU 熵的关系（全部负载，p<0.001）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	–0.02(p=0.87)	0.07(p=0.54)	**–0.48**
有效熵	0.87	0.61	**–0.60**
正则化的使用熵	–0.08(p=0.47)	0.15(p=0.17)	**–0.33**
正则化的有效熵	0.81	0.40	**–0.83**

表 8 单核通量与 CPU 熵的关系（全部负载，p<0.001）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.10(p=0.37)	0.26(p=0.012)	**–0.43**
有效熵	0.90	0.55	–0.14(p=0.16)
正则化的使用熵	–0.15(p=0.16)	0.29(p=0.0057)	**–0.36**
正则化的有效熵	0.85	0.30(p=0.0041)	**–0.43**

铁原型分别在 98.1%、100%的样本中具有更低的正则化的 CPU 使用熵和有效熵. 正则化的 CPU 使用熵、有效熵分别在 98.1%、77.78%的样本中降低 50%，在 92.6%、3.7%的样本中降低 75%. 在加速度为 10、任务峰值为 10 的情况下，K8s-4C8C 具有更低的正则化后的 CPU 使用熵，是因为该负载恰好适合 K8s- 4C8C 的配置执行，使用率接近 100%，良率也接近 100%.

表5~表8表明良率和单核通量均与正则化的CPU使用熵呈现显著的中等程度负相关，良率和单核通量分别与 CPU 有效熵呈现显著的强和中等程度负相关. 该结果也支持了低熵假设.

5.4 优先化假设验证

本小节分别从低优先级任务对良率影响、资源分配使用和结果到达顺序三个方面对优先化假设进行验证. 实验结果表明，信息高铁原型可以在基础设施层提供优先化的能力. 也就是说，用户无须在应用层自行构建的高低优先级任务的调度和资源分配，只需向信息高铁原型传递优先化标签即可保证高优先级任务具有资源的优先获取权且被更早的执行.

在执行测试任务阶段，本文通过测试高低优先级负载，验证信息高铁原型的优先化. 高低优先级负载与渐变负载类似，但是物端在发送一个高优先级任务后，立即发送一个低优先级的任务. 高低优先级负载由高优先级任务的加速度和任务峰值唯一确定. K8s 的低优先级任务将会发送给特定容器（低优先级容器），该容器的配置为 K8s-4C8C 且初始在北京南京各部署 1 个容器.

实验通过比较渐变负载和高低优先级负载的良率来验证系统的优先化能力. 相同参数下，高低优先级负载中的高优先级任务的良率相比渐变负载下降的越少，说明系统的优先化能力越强. 在加速度为 0.2、任务峰值为 50 的情况下，与渐变负载相比，K8s-1C2C、K8s-4C8C、信息高铁原型的良率分别下降 59%、30.7%和 0.6%. 在加速度为 1、任务峰值为 130 的情况下，与渐变负载相比，K8s-1C2C、K8s-4C8C、信息高铁原型的良率分别下降 99.6%、68.0%和 8.5%. 选取上述测试点的依据是：在加速度为 0.2 时，K8s-1C2C、K8s-4C8C 的良率拐点出现在任务峰值为 50 的时候；加速度为 1 时，信息高铁原型的良率拐点出现在任务峰值为 130 的时候. 实验结果表明，在信息高铁原型中，低优先级任务对高优先级任务的良率影响更小，支持了优先化假设.

本文详细分析了两类系统在执行任务峰值170，加速度 50 的高低优先级负载时 CPU 使用情况，以验证信息高铁原型在资源分配方面的优先化. K8s 系统采用 K8s-4C8C 配置和极端的容器配置——K8s-10mC20C，即请求 10 毫核+上限 20 核+60%扩容. 低优先级容器初始在南京和北京各部署 8 个容器且南京北京每台服务器上各绑定 20 核 CPU 给 K8s 系统使用. K8s 系统的实验结果如图 12(a)、图 12(b)所示，K8s-10mC20C 系统中的高优先级任务获取 CPU 核数不足 0.5 核，K8s-4C8C 系统中高优先级任务获取 4 核，均远小于低优先级任务获取的 CPU 资源. 图 12(c)显示，信息高铁原型中高优先级

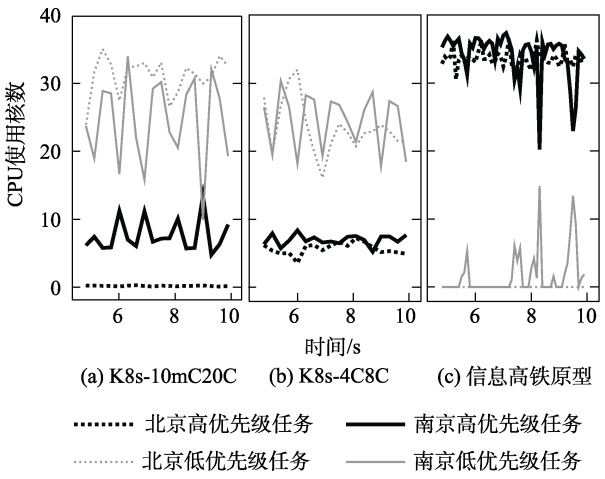


图 12 信息高铁原型和 K8s 的 CPU 使用

任务获得的 CPU 核数则是低优先级任务的 12.2 倍，说明信息高铁原型通过任务的优先级标签可以保证高优先级任务优先获取 CPU 资源.

在执行红包任务阶段，通过任务结果到达顺序，验证信息高铁原型实现了独占用户具有最高优先级，高级用户具有次高优先级. 三类角色在收到“开通通知”后会向开发团队发送总共 322 个红包任务，细节参见 4.3 节. 实验模拟 NTP 授时会出现时间偏差的现象，最坏情况下时间偏差会超过 100 ms[32]. 我们将物端的本地时间分别向前漂移 30 ms、50 ms 和 70 ms，系统其他部分时间保持不变. 独占用户和高优先及用户任务完成情况如图 13 所示.

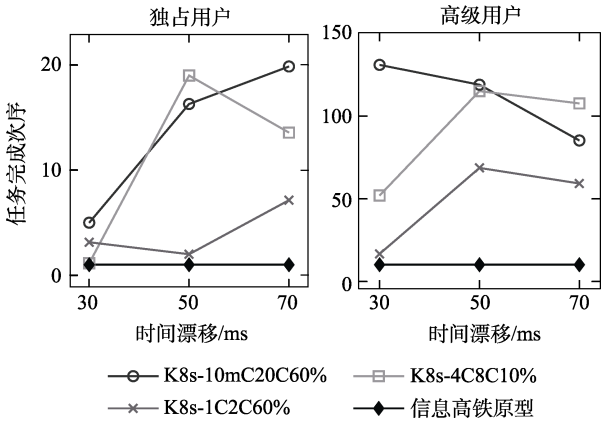


图 13　独占和高级用户红包任务完成次序

实验结果显示：（1）信息高铁原型中，独占用户的任务每一次均最先完成，而 K8s 系统均未实现. （2）信息高铁原型中，高级用户的红包任务靠前完成，平均完成次序为前 3.1%，而 K8s 系统中高级用户任务的平均完成次序为前 25.8%. 实验结果在一定程度上验证了依靠全局一致时钟的信息高铁原型可以实现基于物理时间的独占使用，并在基础设施层提供不同优先级的支持. 但是由于缺乏网络层的优先化支持，仍有可能出现独占用户任务不能最先完成的情况. 未来工作会采用专用且支持优先化的网络来解决该问题.

5.5　三点其他观察

信息高铁原型展示了同时实现高良率和高 CPU 使用率的能力. 图 14 给出了在加速度为 0.2 的渐变负载下，各系统的 CPU 使用率. 实验结果显示，在 84%的样本中，信息高铁原型的 CPU 平均使用率在 90%以上，平均使用率最低为 74.7%. K8s 系统的 CPU 平均使用率在 1.2%~100%之间，且与其具体配置相关. 相关性分析显示，信息高铁原型的 CPU 平均使用率与良率的 Spearman 相关系数为 0.64(p< 0.001)，是正相关. 该结果说明通过低熵有序的资源使用，信息高铁有望实现在不降低服务品质的前提下达到高使用率.

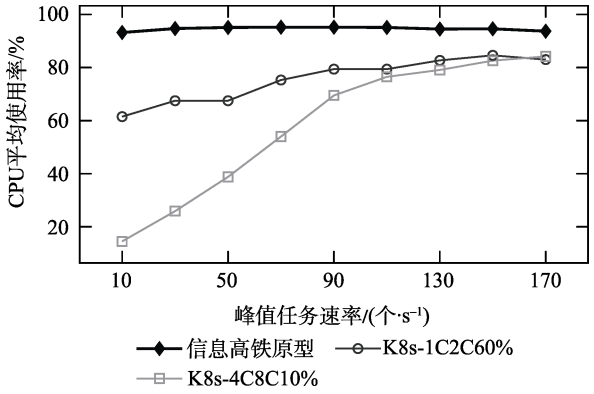


图 14　CPU 平均使用率（加速度=0.2）

信息高铁原型具有更强的负载变化适应能力. 从图 9 可以得出，信息高铁原型可以在保持高良率的情况下，应对加速度大于 10 的渐变负载；而 K8s-1C2C 和 K8s-4C8C 系统在保持通量情况下，只可应对加速度低于 2 的负载变化. 因此，相比 K8s 常见配置，信息高铁原型的负载变化适应能力至少提升了 5 倍.

信息高铁原型可以缓解 CPU 资源的占而不用问题. K8s 系统由于 CPU 配额的不精确性和缩扩容的滞后性会导致 CPU 资源的占而不用. 例如在任务峰值为 50、加速度为 1 的渐变负载下，大约在 18 秒后，K8s-4C8C 系统的 CPU 使用核数为配额核数的 0~74%，大多数情况下低于 50%. 这段时间内未被实际使用的 CPU 配额核数将不能分配给新的容器. 信息高铁原型要求任务提供精确的 CPU 核数需求和不精确的运行时长估计，并根据上述参数分配 CPU 核数和时长. 如果任务提前结束，则执行机会立即回收剩余 CPU 核时，并通过毫秒级资源上报机制使得站点及时获取执行机最新资源状态，进而调度新任务给执行机. 通过上述技术，信息高铁原型的 CPU 分配（配额）与使用基本一致，缓解了 CPU 资源占而不用的问题.

6　结　论

本文目的是对信息高铁的低熵高通量高品质潜力提供实验数据支撑. 我们构建了基于广域一致时钟的信息高铁原型系统，并与基于 Kubernetes 的云计算系统进行对比. 实验设计了物端采集-云端计算的分布式智能负载，包含突发和渐变两种特征，并设计了红包任务来验证系统的优先化能力.

为了定量地刻画低熵高通量高品质性质，我们量化定义了通量、良率与两种 CPU 熵. 通过与 K8s 云计算实验系统的对比，本工作初步验证了信息高铁的高通量、低熵和优先化三个假设，并得出三点具体结论：

第一，在采用同样裸资源处理相同负载的条件下，信息高铁原型数量级提升了良率和通量. 在突发负载、渐变负载两种测试场景下，相比 K8s 系统，信息高铁原型的良率分别在 55.5%、12.2%的样本中至少提升 1 倍和 1000 倍，通量分别在 58.9%、14.4%的样本中至少提升 1 倍和 1000 倍.

第二，信息高铁原型的良率和单核通量均与 CPU 熵呈现显著负相关. 相比 K8s 系统，信息高铁原型的正则化后的 CPU 熵呈现明显下降趋势，在超过 90%的样本中，降至 K8s 系统的 50%以下. 因此，追求低熵有序是实现信息高铁的高通量高良率目标的可行方向.

第三，信息高铁原型在基础设施层提供优先化能力，自动实现了高低优先级任务调度和资源分配. 信息高铁原型在基础设施层提供多种优先级调度策略，保障了高优先级任务的服务质量；并通过全局一致时钟，支持了基于物理时间的跨域独占优先级任务执行.

从实验数据分析可初步得到三点观察：

第一，信息高铁原型可在保持高良率的情况下实现高 CPU 使用率. 在 84%的样本中，信息高铁原型的 CPU 平均使用率在 90%以上.

第二，信息高铁原型可适应更强的负载变化并保持高良率. 在保持高良率的前提下，K8s 系统仅可应对每个物端每秒增加 0~2 个任务，而信息高铁原型可应对每个物端每秒增加 0~10 个任务.

第三，信息高铁原型可缓解 CPU 资源占而不用的问题. 原型系统实现了 CPU 分配和使用的基本一致，可避免 K8s 系统因配额核数不准确和缩扩容的滞后性导致的 CPU 资源占而不用.

本文报告了信息高铁研究工作的第一个现场实验结果，采用的应用负载较为同构单一且实验资源总量有限. 未来将使用大数据分析、实时在线服务、人机物三元计算等多种类型的应用负载，在更大规模的跨域分布式资源上验证信息高铁是否仍可保持高通量、高良率和低熵的性质.

致 谢 感谢李国杰院士和孙凝晖院士的指导、反馈与鼓励. 感谢王晓虹老师在工作讨论中给予的建议和启发. 中科南京信息高铁研究院提供了实验场地和环境帮助，特此致谢！

参 考 文 献

[1] Sun Ninghui. Thoughts on new IT technique system. Bulletin of Chinese Academy of Sciences, 2022, 37(1): 8-14 (in Chinese)
(孙凝晖. 对信息技术新体系的思考. 中国科学院院刊, 2022, 37(1): 8-14)

[2] Wang Xiaohong, Wang Sa, Tang Hongwei, Peng Xiaohui. Building substrate for national strategy of new infrastructure construction—practice and thought of information superbahn testbed. Bulletin of Chinese Academy of Sciences, 2021, (9): 1066-1073 (in Chinese)
(王晓虹, 王卅, 唐宏伟, 彭晓晖. 构建“新基建”国家战略的技术底座——“信息高铁”综合试验场建设的实践与思考. 中国科学院院刊, 2021, 36(9): 1066-1073)

[3] Xu Zhiwei, Li Guojie, Sun Ninghui. Information superbahn: towards new type of cyberinfrastructure. Bulletin of Chinese Academy of Sciences, 2022, 37(1): 46-52 (in Chinese)
(徐志伟, 李国杰, 孙凝晖. 一种新型信息基础设施：高通量低熵算力网（信息高铁）. 中国科学院院刊, 2022, 37(1): 46-52)

[4] Xu Zhiwei, Li Chundian. Low-entropy cloud computing systems. SCIENTIA SINICA Informationis, 2017, 47(9): 1149-1163 (in Chinese)
(徐志伟, 李春典. 低熵云计算系统. 中国科学: 信息科学, 2017, 47(9): 1149-1163)

[5] Verma A, Korupolu M, Wilkes J. Evaluating job packing in warehouse-scale computing//2014 IEEE International Conference on Cluster Computing. Madrid, Spain, 2014: 48-56

[6] Vahdat A. Networking challenges for the next decade//Google Networking Research Summit Keynote Talks. Santa Clara, USA, 2017

[7] Verma A, Pedrosa L, Korupolu M, et al. Large-scale cluster management at google with borg//Proceedings of the Tenth European Conference on Computer Systems. New York, USA, 2015: 1-17

[8] Schwarzkopf M, Konwinski A, Abd-El-Malek M, et al. Omega: flexible, scalable schedulers for large compute clusters//Proceedings of the 8th ACM European Conference on Computer Systems. New York, USA, 2013: 351-364

[9] Burns B, Grant B, Oppenheimer D, et al. Borg, Omega, and Kubernetes: lessons learned from three container-management systems over a decade. Queue, 2016, 14(1): 70-93

[10] Medel V, Rana O, Bañares J Á, et al. Modelling performance & resource management in kubernetes//Proceedings of the 9th International Conference on Utility and Cloud Computing. Shanghai, China, 2016: 257-262

[11] Kjorveziroski V, Filiposka S. Kubernetes distributions for the edge: serverless performance evaluation. The Journal of Supercomputing, 2022, 78(11): 13728-13755

[12] Changpeng Zhu, Bo Han, and Yinliang Zhao. 2022. A comparative performance study of spark on kubernetes. The Journal of Supercomputing. 2022, 78(11): 13298-13322

[13] Xu Zhiwei, Bao Yungang, Xiong Jin, et al. Achievements of study on the theory and method of software defined cloud computing. China Basic Science, 2020, 22(01): 49-53

(徐志伟, 包云岗, 熊劲, 向东等. 软件定义的云计算基础理论与方法的研究进展. 中国基础科学, 2020, 22(01): 49-53)
[14] Dean, Jeffrey, Luiz André Barroso. The tail at scale. Communications of the ACM, 2013, 56(2): 74-80
[15] Hellerstein J, Faleiro J, Gonzalez J, et al. Serverless computing: one step forward, two steps back//9th Biennial Conference on Innovative Data Systems Research. Asilomar, USA, 2019: 1-9
[16] Jonas E, Schleier-Smith J, Sreekanti V, et al. Cloud programming simplified: a Berkeley view on serverless computing. Technical Report: UCB/EECS-2019-3, EECS Department, University of California, Berkeley, USA, 2019
[17] Baldini I, Castro P, Chang K, et al. Research advances in cloud computing. Singapore: Springer Nature, 2017
[18] Guo J, Chang Z, Wang S, et al. Who limits the resource efficiency of my datacenter: an analysis of alibaba datacenter traces//2019 IEEE/ACM 27th International Symposium on Quality of Service. Arizona, USA, 2019: 1-10
[19] Yang Y, Kong X, Zhao L, et al. SDCBench: a benchmark suite for workload colocation and evaluation in datacenters. Intelligent Computing, 2022, 2022: 1-18
[20] Zhi-Wei Xu, Zhen-Ying Li, Zi-Shu Yu, Feng-Zhi Li. Information superbahn: Towards a planet-scale, low-entropy and high-goodput computing utility. Journal of Computer Science and Technology, 2023, 38(1): 103-114
[21] Shannon C E. A mathematical theory of communication. The Bell System Technical Journal, 1948, 27(3): 379-423
[22] Alajaji F, Chen P N. An Introduction to Single-User Information Theory. Singapore: Springer Nature, 2018
[23] Pillonetto G, Chen T, Chiuso A, et al. Regularized system identification: learning dynamic models from data. Cham: Springer Nature, 2022
[24] Hu Z, Li B, Luo J. Time-and cost-efficient task scheduling across geo-distributed data centers. IEEE Transactions on Parallel and Distributed Systems, 2017, 29(3): 705-718
[25] Li C, Zhang Y, Hao Z, et al. An effective scheduling strategy based on hypergraph partition in geographically distributed datacenters. Computer Networks, 2020, 170: 107096
[26] Lee J, Hwang K. YOLO with adaptive frame control for real-time object detection applications. Multimedia Tools and Applications, 2022, 81(25): 36375-36396
[27] Zhang Y, Yu Y, Wang W, et al. Workload consolidation in alibaba clusters: the good, the bad, and the ugly//Proceedings of the 13th Symposium on Cloud Computing. San Francisco, USA, 2022: 210-225
[28] Pons L, Feliu J, Sahuquillo J, et al. Cloud white: detecting and estimating QoS degradation of latency-critical workloads in the public cloud. Future Generation Computer Systems, 2023, 138: 13-25
[29] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets//2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10). Boston, USA, 2010:10
[30] Spearman C. The proof and measurement of association between two things. The American Journal of Psychology, 1904,15(1): 72-101
[31] Haldun Akoglu. User's guide to correlation coefficients. Turkish Journal of Emergency Medicine, 2018, 18(3): 91-93
[32] Mills D L. Computer Network Time Synchronization: The Network Time Protocol on Earth and in Space. 2nd. USA: CRC Press, 2010

附录A

该附录回答下列问题：

选取不同的预估峰值 CPU 资源需求占总 CPU 资源的比例作为区分高低负载的阈值是否会影响本文验证高通量假设、低熵假设和优先化假设的结果？

定义 7. 划分高低负载的阈值 TH. 定义阈值 TH=x 为：将预估峰值 CPU 资源需求占总 CPU 资源的比例为 x 作为区分高低负载的分界线.

正文中选取的 TH=25%.

选取 25%~50%中的其他阈值，高通量假设依然成立. 观察图 9 可知，在超过 25%阈值的负载中，在大多数情况下，信息高铁原型相对 K8s 系统的性能提升可达数倍.

选取 25%~50%中的其他阈值，低熵假设依然成立. 表 9~表 20 给出了 TH 等于 25%到 50%时，高负载情况下 CPU 两种熵与系统良率和通量的 Spearman 相关系数. 结果表明在上述 6 种阈值条件下，信息高铁原型系统的良率和单核通量均与 CPU 熵呈现显著的负相关，可以得出与正文中一致的结论. 因此我们认为，正文中选取 TH=25%不会影响正文中低熵假设的验证结果.

选取其他阈值，优先化假设依然成立. 因为优先化假设的验证并不依赖于区分高低负载.

表 9 良率与 CPU 熵的关系（高负载，p<0.001，TH=25%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.30(p=0.026)	0.54	**–0.71**
有效熵	0.94	0.81	**–0.86**
正则化的使用熵	–0.65	–0.04(p=0.78)	**–0.50**
正则化的有效熵	0.91	0.72	**–0.92**

表 10 良率与 CPU 熵的关系（高负载，p<0.001，TH=30%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.30(p=0.035)	0.42(p=0.002)	**–0.72**
有效熵	0.95	0.84	**–0.84**
正则化的使用熵	–0.66	–0.27(p=0.054)	**–0.53**
正则化的有效熵	0.94	0.75	**–0.92**

表 11 良率与 CPU 熵的关系（高负载，p<0.001，TH=35%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.32(p=0.026)	0.44(p=0.002)	**–0.74**
有效熵	0.96	0.83	**–0.84**
正则化的使用熵	–0.65	–0.26(p=0.079)	**–0.53**
正则化的有效熵	0.94	0.74	**–0.92**

表 12 良率与 CPU 熵的关系（高负载，p<0.001，TH=40%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.29(p=0.08)	0.27(p=0.099)	**–0.75**
有效熵	0.95	0.89	**–0.82**
正则化的使用熵	–0.62	–0.52	**–0.52**
正则化的有效熵	0.95	0.76	**–0.90**

表 13 良率与 CPU 熵的关系（高负载，p<0.001，TH=45%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.25(p=0.15)	0.29(p=0.091)	**–0.75**
有效熵	0.95	0.88	**–0.81**
正则化的使用熵	–0.61	–0.50(p=0.002)	**–0.51**
正则化的有效熵	0.95	0.73	**–0.89**

表 14 良率与 CPU 熵的关系（高负载，p<0.001，TH=50%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.28(p=0.14)	–0.05(p=0.8)	**–0.68**
有效熵	0.96	0.91	**–0.75**
正则化的使用熵	–0.55(p=0.002)	–0.66	**–0.48**
正则化的有效熵	0.96	0.82	**–0.86**

表 15 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 25%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.33(p=0.014)	0.37(p=0.005)	**–0.58**
有效熵	0.92	0.58	**–0.60**
正则化的使用熵	–0.61	0.11(p=0.43)	**–0.38**
正则化的有效熵	0.88	0.48	**–0.55**

表 16 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 30%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.32(p=0.021)	0.26(p=0.071)	**–0.61**
有效熵	0.94	0.63	**–0.69**
正则化的使用熵	–0.63	–0.09(p=0.55)	**–0.41**
正则化的有效熵	0.92	0.53	**–0.60**

表 17 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 35%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.34(p=0.017)	0.30(p=0.037)	**–0.61**
有效熵	0.94	0.68	**–0.70**
正则化的使用熵	–0.62	–0.12(p=0.4)	**–0.41**
正则化的有效熵	0.92	0.58	**–0.60**

表 18 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 40%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.30(p=0.063)	0.17(p=0.3)	**–0.64**
有效熵	0.93	0.80	**–0.75**
正则化的使用熵	–0.58	–0.42(p=0.009)	**–0.45**
正则化的有效熵	0.92	0.69	**–0.66**

表 19 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 45%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.27(p=0.12)	0.18(p=0.3)	**–0.64**
有效熵	0.93	0.78	**–0.75**
正则化的使用熵	–0.56	–0.38(p=0.021)	**–0.44**
正则化的有效熵	0.91	0.66	**–0.66**

表 20 单核通量与 CPU 熵的关系（高负载，p<0.001，TH= 50%）

Spearman 相关性	1C2C	4C8C	信息高铁原型
使用熵	0.28(p=0.13)	–0.14(p=0.46)	**–0.66**
有效熵	0.93	0.89	**–0.77**
正则化的使用熵	–0.51(p=0.004)	–0.66	**–0.47**
正则化的有效熵	0.93	0.84	**–0.69**

YU Zi-Shu, Ph.D. candidate. His research interests include distributed systems and runtime management.

LI Feng-Zhi, Ph.D. candidate. His research interests include distributed systems and causal science.

ZHENG Shou-Jian, M.S. candidate. His main research interests include edge computing and resource management.

WANG Yi-Fan, Ph.D., assistant professor. His main research interests include distributed systems and resource management.

ZHANG Xing-Zhou, Ph.D., assistant professor. His main research interests include distributed computing systems and programming methods.

PENG Xiao-Hui, Ph.D., associate professor. His main research interests include distributed systems and operating systems.

XU Zhi-Wei, Ph.D., professor. His main interests include computer architecture and distributed systems.

Background

Computing technology is entering a new era of the Intelligent Internet of Everything. In this era, information infrastructure needs to handle a massive number of tasks with Quality-of-Service (QoS) requirements from trillions of smart devices, while achieving high efficiency. A new measurement called “goodput” is used to show how many QoS-satisfied tasks can be processed in a given amount of time.

Information Superbahn was proposed to meet the requirements of this new era. It is designed as a planet-scale, high-goodput, low-entropy computing network. However, there is a lack of empirical evidence to verify the hypothesis that high-goodput can be achieved by decreasing entropy in computing systems like Information Superbahn. The purpose of our work is to provide such evidence.

A large-scale testbed has recently been established between Beijing and Nanjing to evaluate the Information Superbahn prototype and associated technologies. We constructed an Information Superbahn prototype system on this testbed to verify our hypotheses. In comparison to a computing system based on Kubernetes, our research demonstrates that Information Superbahn can substantially improve yield and goodput by orders of magnitude, given the same workload and resources. We also define and measure CPU entropy, which exhibits a significant negative correlation with goodput, indicating that a low-entropy approach has the potential to improve the system's yield (the percentage of QoS satisfied tasks) and goodput. Our work supports the vision of Information Superbahn and demonstrates its potential to significantly increase goodput and yield by reducing entropy.

The research is supported by the Nation Natural Science Foundation of China (No.62072434 and No. U19B2024), the Innovation Fund from the Institute of Computing Technology, Chinese Academy of Sciences (No. E261010).