智能移动终端计算迁移研究

张文丽[1),2)] 郭兵[1)] 沈艳[3)] 王毅[1)] 熊伟[1)] 段林涛[4)]

[1)](四川大学 计算机学院，成都 610065)

[2)](成都工业学院 计算机工程学院，成都 611730)

[3)](成都信息工程学院 控制工程学院，成都 610225)

[4)](成都大学 信息科学与技术学院，成都 610106)

摘 要 随着智能移动终端的普及和移动应用对计算资源需求的不断增长，移动终端资源（包括计算、存储、能量等）受限问题日益突出，如何扩展移动终端资源成为移动计算领域需要迫切解决的问题。计算迁移(computation offloading)是解决移动终端资源受限问题的一个有效途径，通过将一部分计算任务从本地迁移到远程设备执行来扩展移动终端的资源。本文首先回顾了计算迁移在分布式计算、普适计算和云计算背景下具有代表性的研究工作和进展，然后具体介绍和分析了三个典型的计算迁移系统，在此基础上，从软件架构角度对计算迁移系统的内部组成结构和关键质量属性等共性问题进行了探讨，并试着提出了计算迁移系统的参考架构。最后，探讨未来计算迁移的研究挑战和发展趋势。

关键词 计算迁移；智能移动终端；划分；移动云计算；移动增强；

中图法分类号 TP338

Computation Offloading on Intelligent Mobile Terminal

Zhang Wen-Li[1),2)] Guo Bing[1)] Shen Yan[3)] Wang Yi[1)] Xiong Wei[1)] Duan Lin-Tao[4)]

[1)](Department of Computer Science, Sichuan University, Chengdu 610065)

[2)](Department of Computer Engineering, Chengdu Technological University, Chengdu 611730)

[3)](Department of Control Engineering, Chengdu University of Information Technology, Chengdu 610225)

[4)](School of Information Science and Technology, Chengdu University, Chengdu 610106)

Abstract With the pervasive usage of the intelligent mobile terminals and the applications' ever-increasing requirement for the resources, the problem of the resources (include computation, storage and energy) limitation on the mobile terminals becomes increasingly distinct. How to extend the resources of the mobile terminals has become the problem which needs to be solved urgently in the field of mobile computing. Computation offloading has been shown to be an effective approach to solve this problem, which means sending parts of the computation from local to remote devices to extend the resources of the mobile terminals. This paper reviews the representative works on the computation offloading in the context of three computation models: distributed computation, pervasive computation and cloud computation, then introduces and analyzes three classic computation offloading systems in detail. Based on these work, we explore the common problems about the structure of the computation offloading systems and their key quality attributes from the view of software

本课题得到国家自然科学基金重点项目(61332001）、国家自然科学基金项目(61272104，61472050)资助.张文丽，女，1979年生，博士研究生，讲师，主要研究领域为嵌入式实时系统、绿色计算，E-mail: zhangwenli2007@gmail.com.郭兵（通信作者），男，1970年生，博士，教授，博士生导师，CCF高级会员,主要研究领域为嵌入式实时系统、绿色计算，E-mail: guobing@scu.edu.cn. 沈艳，女，1973年生，博士，副教授，主要研究领域为智能化网络化测控技术、智能仪器，E-mail: shenyan02@163.com.王毅，男，1976年生，博士研究生，讲师，主要研究领域为嵌入式实时系统，E-mail:wykm76@126.com.熊伟，男，1979年生，博士研究生，讲师，主要研究领域为嵌入式实时系统，E-mail:xwedu79@163.com. 段林涛，男，1977年生，博士研究生，讲师，主要研究领域为嵌入式实时系统，E-mail: duanlintao@cdu.edu.cn.

architecture, trying to present the reference architecture of the computation offloading systems. At last, this paper discusses the future research challenges and the development tendency.

Key words computation offloading; intelligent mobile terminal; partition; mobile cloud computing; mobile augmentation;

1 引言

过去十年，全球移动用户数量经历了飞速增长，GSMA(Global System for Mobile Communications Association，全球移动通讯系统协会)在 2014 年移动经济报告①中指出，截止 2013 年，全球移动设备联网数达到 69 亿，预计到 2020 年，还将增加 40 亿。目前，智能移动终端已成为最流行的计算平台，其性能正变得越来越强大，高端移动终端通常配备有高速无线接口、G 比特内存容量、GHz 速度的处理器、以及各种传感器，以至于用户对移动终端的期望越来越高——他们希望能够在移动终端上运行 PC 机上的应用。为了迎合用户需求，开发人员也正在开发功能更加复杂的移动应用，例如富媒体应用、增强现实、自然语言处理等，但事实上，移动终端很难运行这些应用，因为它们的资源有限。在本文中，移动终端的资源包括计算、存储和能量等资源。

移动终端资源受限，是由移动性决定的[1]。由于尺寸和重量受到约束，移动终端在处理能力、内存容量、网络连接和电池容量等方面必然受到限制。此外，虽然近年来移动终端的处理速度和存储容量已得到极大增长，但电池容量的增长却十分缓慢——每年的增长速度只有5%②。IDC（International Data Corporation，国际数据公司）对25个国家的50,000个智能手机用户进行研究，在2014年的“智能手机使用和购买驱动力”报告③中指出：在购买智能手机时，用户认为待机时间比任何其他特性（包括易用性、屏幕大小、品牌、相机分辨率等）都重要。资源受限问题已经成为移动终端和移动应用进一步增值的瓶颈。因此，如何扩展移动终端的资源，成为需要迫切解决的问题。

计算迁移是解决移动终端资源受限问题的一个有效途径。它将应用中资源使用密集的计算任务从本地发送到远程设备执行，通过利用远程资源来扩展本地资源。

根据远程设备和移动终端物理距离的远近关系，将远程设备分为两类，一类是近距离设备，包括 surrogate（代理）、cloudlet（薄云）或移动终端周围其他的移动设备，另一类是远距离设备，包括云服务器。它们对应着四种计算迁移的实现方法，即基于 surrogate、基于 cloudlet、基于移动设备和基于云。对应关系如图 1 所示。本文将在第 2 节描述计算迁移的四种实现方法。

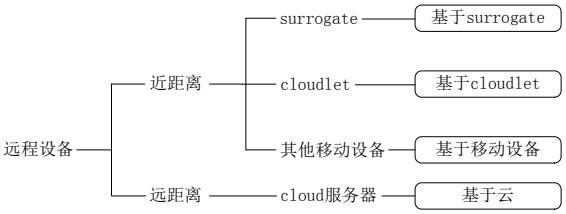

图 1 计算迁移实现方法和计算资源的对应关系

C/S 架构和计算迁移在方法上有相似之处，都是将应用划分到本地和远程服务器上执行。它们的区别如表 1 所示。和 C/S 架构相比，计算迁移具有更好的灵活性，能够更好地适应应用执行条件的变化。

表 1 C/S 架构与计算迁移的比较

概念	划分粒度	划分时机	目标	拓扑结构
计算迁移	方法、对象、类	动态划分	扩展资源	多个节点
C/S 架构	事务逻辑	静态划分	降低系统通讯开销	客户端 服务器

计算迁移能够为移动用户、开发人员、手机生产商和移动运营商带来利益，如图 2 所示。从移动用户角度来看，计算迁移能够使移动用户在低端移动终端上运行复杂应用（例如计算密集型应用和存储密集型应用），提升和丰富用户体验，为用户节约金钱成本；从应用开发人员的角度来看，计算迁移系统使移动应用开发人员在开发过程中将更多的精力投入到应用本身，而不必花太多精力考虑移动平台异构性、硬件性能差异性和资源限制等问题；从手机生产商和移动运营商的角度来看，计算迁移技术可能为移动设备和移动应用带来新的增值机会。

① http://www.gsmamobileeconomy.com/

② https://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=4645

③ http://www.idc.com/getdoc.jsp?containerId=247154

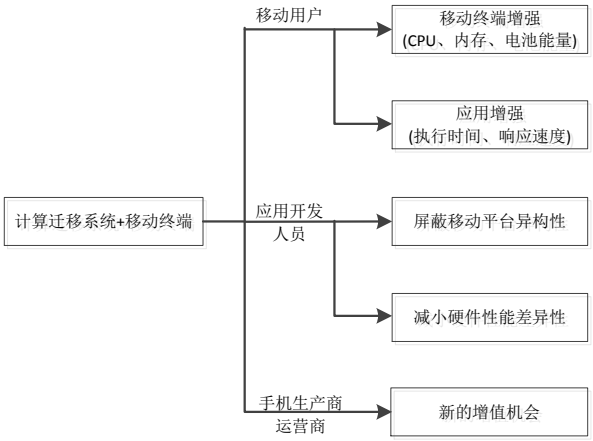

图 2 计算迁移系统为用户/商家带来的利益

本文对计算迁移的发展过程、计算迁移系统的组成结构及关键的质量评价指标进行综述，主要分为以下六个部分：第 2 节回顾计算迁移在三个不同背景下具有代表性的研究工作和待解决的问题；第 3 节具体介绍和分析三个经典的计算迁移系统；第 4 节和第 5 节分别从软件架构角度总结一般计算迁移系统的内部组成结构和关键的质量评价指标；第 6 节试着提出计算迁移系统的参考架构；最后对全文进行总结，提出未来的研究需求和发展趋势。

2 计算迁移的发展

过去十几年，已有大量计算迁移相关的研究工作。本文按照不同背景，即分布式计算、普适计算和云计算，将这些工作分为三个阶段。从研究动机、研究内容和实现方法三个方面，对三个阶段的研究工作进行比较，如表 2 所示。

表 2 计算迁移的发展

背景	时间	研究动机		研究内容	实现方法
		移动终端	应用		
分布式计算	1995-2000	便携式计算机的性能/能耗	计算密集型应用	可行性	开发人员手动划分
普适计算	2001-2008	便携式计算机和智能手机的性能/能耗	资源密集性应用（多媒体应用、图像处理）	系统架构与实现	基于 surrogate
云计算	2009 至今	移动设备增强	富移动应用、增强现实	基于云的计算迁移系统、弹性、可扩展性、可用性	基于 cloudlet/cloud/移动设备

2.1 分布式计算阶段

我们将计算迁移在 1997-2000 年之间研究工作划分到分布式计算阶段。由于当时无线网络和移动设备技术不够发达，因此，移动终端面临的问题主要是便携式计算机的性能较差和待机时间较短等问题。计算迁移在该阶段的研究工作主要是针对便携式计算机，以提升应用性能和节能为目标，探讨和验证远程执行可行性。本阶段产生了三个重要理念，为计算迁移奠定了理论和实践基础：

（1）使用软件方法降低移动计算机的 CPU 能耗[2]。该方法借用分布式系统负载共享的概念，它先计算出任务在本地和基站的执行时间，并对响应时间进行评估，再进一步决定是否将任务迁移到基站执行；

（2）自动生成分布式应用[3]。Hunt G C 和 Scott M L 认为，应用分解任务应该由系统软件而不是应用开发人员来完成。为了验证这个想法，作者开发了名为 Coign 的分布式应用自动划分系统，它是第一个能够自动划分和部署二进制应用的系统，极大地减轻了应用开发人员的负担。该工作为自动应用划分技术奠定了重要基础；

（3）远程执行。Rudenko A 和 Reiher P 在文献[4]首次探讨和验证了通过远程执行节约便携式计算机能耗的可行性。作者进行了一系列实验，对程序在本地和远程执行的能耗进行了比较，验证结果表明，远程执行能否为设备节约能耗，取决于通信能耗和本地处理能耗的折中，此外，客户端采用的节能技术将会影响节能效果。

在分布式计算阶段，有关工作尚且缺乏系统架构和具体实现方面的研究成果，但是为计算迁移奠定了重要的理论基础。

2.2 普适计算阶段

随着硬件技术的发展，普适计算[5]的关键要素（包括手持和可穿戴设备、无线局域网、带有传感

和控制装置的设备等）逐渐成为商业化产品，普适计算开始兴起。智能移动终端的普及应用和性能的不断提高，使得移动应用对资源的需求也不断增长，资源密集型移动应用开始出现，例如语音识别、图像处理等，这导致移动终端硬件资源受限，特别是待机时间较短的问题变得日益突出，于是计算迁移的研究重心开始从便携式计算机转移到智能移动终端，如何扩展资源，提升应用性能，特别是延长待机时间，成为计算迁移在普适计算阶段需要迫切解决的问题。该阶段出现了大量关于计算迁移架构和实现方面的研究成果。Aura[6]和 MAUI[7]是最具代表性和启发意义的两个研究成果。

Aura 是卡内基梅隆大学一个著名的普适计算项目，作者 Satyanarayanan 在文献[6]对它进行了介绍，分析了普适计算给计算机系统研究领域带来的挑战，分别从用户意图、自适应、客户端功能设计、环境感知、前瞻性和透明性等方面对普适计算面临的问题进行探讨，并首次提出 cyber foraging（游牧服务）的概念。

cyber foraging 指：通过利用有线硬件基础设施动态地扩展无线移动计算机的资源。作者将愿意提供计算和存储资源的硬件基础设施称为 surrogate，它指位于公共场合并存在于无线移动计算机周围的台式计算机或者服务器。为了避免网络延迟，cyber foraging 强调 surrogate 与移动计算机在物理位置上邻近。

Aura 客户端架构如图 3 所示。其中，Coda[8]负责远程文件访问，Odyssey/Chroma[9]负责资源监控和自适应，Spectra[10]则负责远程执行。我们重点介绍 Spectra 和 Chroma。

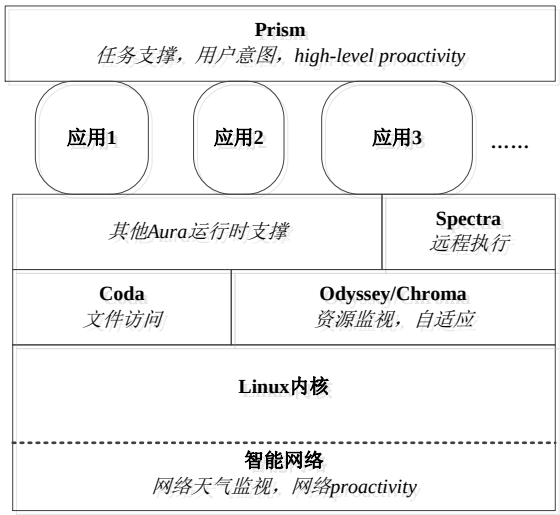

图 3 Aura 客户端架构

作者认为，应该由系统解决普适计算引入的用户意图、Cyber foraging、自适应、环境感知、前瞻性和透明性等问题，而不是把它们留给应用。因此，Spectra 具有两个特性：（1）动态均衡节能、质量和性能目标。Spectra 监视环境条件，根据环境动态地调整节能、性能和应用质量的相对重要性；（2）Spectra 是一个 self-tuning 系统，即 Spectra 不需要应用针对各种各样的平台和输出质量明确定义对应的资源使用需求，而是通过监视和记录应用执行过程中资源使用情况，利用机器学习技术来预测应用将来的资源使用需求。

Chroma 是第一个远程执行系统，包含了 Spectra 的功能。它通过 tactic 来解决自动划分的有效性和无缝性问题。Tactic 指划分计划。Chroma 允许开发人员以简洁声明的形式，为应用中每一个操作定义对应的一个或多个 tactics，其内容包括每个操作对应的资源使用量和保真度级别，开发人员还可以通过 tactic 描述有意义的应用划分。作者将这种以操作为粒度的应用划分称为粗粒度划分。

MAUI 是计算迁移在普适计算阶段另一个具有代表性研究成果，与 Chroma 不同的是，它的目标是通过对应用进行细粒度划分从而最大程度地节能，同时，尽可能减少开发人员修改应用的工作量。MAUI 针对已有的.NET 应用，利用管理代码环境的四个特性（即代码可移植性、反射、类型安全和序列化）来达到上述目的。和 Chroma 一样，MAUI 也为开发人员提供了编程环境，使得开发人员可以对能够被远程执行的方法进行标注。文献[7]的重要贡献和意义在于，它提供了 MAUI 的系统架构及具体实现方法，包括应用划分和迁移执行机制。本文将在第 3 节对 MAUI 进行详细介绍。

本文将 2001 至 2008 年计算迁移相关的工作划分到普适计算阶段，并且将该阶段中计算迁移的实现方法统一归类为基于 surrogate 的计算迁移。对该阶段的文献按照研究内容进行了整理和归类，如表 3 所示。

表 3 普适计算模式下计算迁移相关研究分类

研究内容	相关文献
计算迁移架构	[7][9][10-14]
应用划分算法	[15-16]
资源预测方法	[17-20]
应用开发环境和技术	[21-22]

表 4 从语言依赖性、划分粒度、系统目标等方面对该阶段关键的计算迁移系统进行比较，并简要

介绍它们的特色。

表 4 普适计算阶段关键计算迁移系统的比较

文献	系统名	针对应用	粒度	目标	是否修改源码	特色
[10] [11]	Spectra	C	操作	能耗/响应时间	是	该系统具有自调节性（即不需要预先指定资源使用需求），可根据资源的可用情况及用户偏好动态地调整各系统目标（节能、响应时间等）的相对重要性
[12]	Zap	C	进程	响应时间	否	以一组自包含的进程为划分粒度，该系统解决了之前计算迁移系统存在的迁移过程中进程与进程、进程与操作系统之间相互依赖的问题
[13]	J-Orchestra	Java	对象	自动划分	否	能够自动划分任意 Java 应用，和之前的自动应用划分系统相比，该系统具有更好的通用性、灵活性，自动化程度更高
[14]	Slingshot	Java	组件	响应时间	-	面向已经划分好的应用，解决骨干网高延迟和低带宽导致的应用响应延迟问题，并提出 surrogate 的资源管理方案
[7]	MAUI	C#	方法	能耗	是	面向.NET 应用，之前大多数计算迁移系统的目标是提升应用性能，而该系统的主要目标是节能

计算迁移在普适计算阶段的研究工作主要致力于系统架构及实现，缺乏对安全性、可扩展性以及资源的可用性等问题的考虑，例如，surrogate 的提供者可以在没有预先通知移动用户的情况下，终止 surrogate 的资源共享服务和计算服务，此外，存放在 surrogate 中的数据可以被其他用户访问和修改，这将严重侵犯用户数据的安全和隐私。上述问题很大程度上阻碍了计算迁移系统的实际应用。

2.3 云计算阶段

2008年，云计算概念出现了。云是一种分布式并行系统，由强大的计算机集群组成，用户可以根据他们和服务提供商之间协商的SLA(Service Level Agreement，服务等级协议)，对云资源进行访问[23]。

云资源和surrogate的主要区别在于：

（1）云资源是付费资源，提供商必须保证其质量和可用性，而surrogate是免费资源，提供者不保证能够完成分配的任务；

（2）云资源提供商向用户提供的是虚拟资源，目的是提高资源的利用率和可扩展性，增强用户数据安全性和隐私性，而surrogate缺乏相关方面的考虑。

由于云资源具有更好的可用性、安全性和可扩展性，因此，利用云资源增强移动终端的性能一时间成为颇具吸引力的方法，我们将该方法称为基于云的计算迁移。

然而，基于云的计算迁移方法面临着诸多挑战。由于移动终端通过无线接口与云服务器通信的过程将同时涉及广域网和无线网络，因此，使用云资源必须先解决如下三个问题：

（1）如何避免广域网引入的抖动、错误和时延。通常，云服务器与移动终端的物理距离较远，使用云资源难免会引入抖动、错误和较高的广域网时延（一般为数十毫秒），这将严重降低交互式应用的敏捷度，影响用户体验质量；

（2）如何处理无线网络带宽有限和连接不稳定等问题。之前的大量工作都基于无线网络带宽恒定不变的假设，然而，随着将来越来越多的移动用户开始使用云资源，在高峰时段，网络流量可能急剧增加，导致网络出现延迟和丢包现象，从而严重影响应用性能及用户体验质量；

（3）如何降低使用云资源引入的经济成本。使用云资源的经济成本包括无线通信成本和使用云资源本身带来的成本，然而，目前云资源的使用成本较高。

为了解决上述问题，研究人员提出两种不同的计算迁移实现方法，即基于 cloudlet 和基于移动设备的计算迁移。

文献[1]首次提出利用移动终端附近固定的计算资源增强移动终端的性能。作者将这种资源称为cloudlet，它指与移动终端物理位置邻近的，可信的、资源丰富的、与Internet有着良好连接的计算机或者计算机集群。作者采用动态VM（Virtual Machine，虚拟机）合成来瞬态定制cloudlet上的软件服务，当

附近不存在cloudlet或者cloudlet资源不足时，则使用远程云资源。该过程需要迁移整个VM覆盖，实验结果表明，合成一个VM需要60-90s。文献[24]则采用去重、缩小语义鸿沟、流水线等手段优化VM合成时间。文献[25-28]则从迁移的灵活性、应用性能优化等方面，解决基于cloudlet的计算迁移面临的相关问题。

目前基于 cloudlet 的计算迁移存在的问题有：（1）缺乏统一的 cloudlet 部署方案和管理策略；（2）大多数工作关注 cloudlet 性能和应用性能的优化，缺乏 cloudlet 在信任和安全方面的研究。

基于移动设备的计算迁移在文献[29]中被首次提出，和基于 cloudlet 的计算迁移不同，该方法利用附近其它移动设备上的资源来扩展移动终端资源，这些移动设备通过高速网络(例如 Wi-Fi)相互连接形成 Ad-Hoc 网络，计算任务被划分到该网络中的各个节点上执行。该方法适用于两种场景：（1）位于同一位置区域的多个移动终端相互合作，共同完成同一任务；（2）单个用户或家庭拥有多个移动终端。文献[29]对基于移动设备的计算迁移系统架构进行了初步设计；文献[30-32]分别讨论了具体的计算任务建模方法和调度算法、基于云的 P2P 实现方法、节约整体移动设备能耗的算法等问题。

目前基于移动设备的计算迁移存在如下问题：（1）缺乏统一的资源组织方案和管理策略；（2）缺乏移动设备相互合作所必需的激励和声望系统；（3）缺乏对计算资源的稳定性、安全性以及数据一致性和隐私性方面的研究。

CloneCloud[33-34]是最早出现的基于云的计算迁移系统，通过使用云端资源来增强移动应用的性能、节省移动终端的能耗。文献[34]详细描述了云克隆和应用划分的具体实现方法，对基于云的计算迁移系统的实现具有重要指导意义，该方法的局限性在于，需要预先针对各种执行环境为应用生成划分策略。文献[35-37]探讨了云资源的分配、计算任务的并行执行、移动数据流应用的划分等问题。

目前基于云的计算迁移存在的问题有：（1）严重依赖于高性能网络基础设施。随着富移动应用[38]的产生，基于云的计算迁移系统的通信时延问题将更加突出；（2）用户数据的安全性和隐私性问题；（3）目前大多数相关工作致力于解决应用的能耗、性能和响应时间等问题，缺乏对通信开销和用户经济成本的考虑。

和基于云的计算迁移相近的概念有CMA（Cloud-based mobile augmentation, 基于云的移动增强）[39]MCC[40]（Mobile Cloud Computing，移动云计算）。

CMA是目前最先进的移动增强模型，指通过各种可行方法（包括硬件和软件），增加、增强和优化移动设备计算能力的过程。其中，移动设备实体包括智能手机、平板电脑、手持/可穿戴计算设备和车载计算机。硬件方法涉及到制造高端物理组件，特别是CPU、内存、存储和电池。软件方法包括但不限于计算迁移、远程数据存储、无线通信、资源感知的计算、保真度适应和远程服务请求。可见，本文讨论的计算迁移属于CMA概念中的软件方法。

MCC 目前尚未存在一个被广泛接受的定义，通常指使用移动云资源增强移动终端的一系列技术。对移动云资源的理解不同，将形成不同的 MCC 架构。目前对移动云资源的理解可分为三类：由移动设备组成的虚拟云资源、由第三方（例如 Amazon、Google、IBM 等）提供的公共云资源和由企业或家庭提供的私有云资源。因此，可以把基于 cloudlet 的计算迁移、基于移动设备的计算迁移和基于云的计算迁移看作 MCC 的具体实现方法。

我们对基于 surrogate、基于 cloudlet、基于移动设备和基于云的计算迁移方法进行了比较，如表 5 所示。

表5 四种计算迁移系统实现方法的比较

实现方法	网络延迟	网络类型	计算资源				适用场景
			安全性	可用性	管理策略	收费标准	
基于 surrogate	低	无线	差	低	无	无	针对网络延迟敏感的应用
基于 cloudlet	低	无线	一般	一般	无	无	针对网络延迟敏感的应用
基于移动设备	低	无线	差	低	无	无	针对位于同一位置区域的多台移动设备相互合作、共同完成同一任务的场景
基于云	高	无线和 WAN	好	高	有	有	针对网络延迟不敏感或需要并行处理的应用

本文将 2009 年至今与计算迁移相关的研究工作划分到云计算阶段，对相关文献按照研究内容进

行了整理和分类，如表 6 所示。

表 6 云计算模式下计算迁移相关研究分类

	研究内容	相关文献
实现方法	基于 cloudlet	[1][24-28]
	基于移动设备	[29-32]
	基于云	[33-37]
	基于 surrogate	[41-43]
	划分算法	[44-48]
	资源预测与建模	[49-52]
	应用开发环境、Java 应用重构	[53-56]

对表 6 中关键的计算迁移系统进行简单介绍和比较，如表 7 所示。

表 7 云计算阶段关键计算迁移系统比较

实现方法	文献	系统名	粒度	目标	是否修改源码	特色
基于 cloudlet	[1]	Cloudlet	VM	响应时间	否	首次提出 cloudlet 的概念，支持以 VM 为粒度的应用划分，对系统进行初步设计和实现
	[25] [28]	-	组件	执行时间	-	拓展了文献[1]中 cloudlet 的概念，支持以组件为单位的应用划分，从而使系统更灵活，应用性能也更高，但不支持对已有应用进行划分，而是提供编程模型供开发人员生成新的分布式应用
基于移动设备	[30]	Serendipity	任务	执行时间/能耗	是	提供具体的计算任务建模方法和各种网络连接环境下的计算调度算法
	[31]	Clone2Clone	任务	执行时间/能耗	是	实现了智能手机云克隆的分布式 P2P 平台，目标是实现移动终端之间的通信迁移和计算迁移
基于云	[33] [34]	CloneCloud	线程	执行时间/能耗	否	首次提出 clonecloud 概念，支持对应用层 VM 上的应用进行划分，局限性在于需要针对各种执行环境预先建立划分数据库
	[35]	ThinkAir	方法	执行时间/能耗	是	ThinkAir 与之前计算迁移系统的区别在于，它面向商业云，专注于解决计算资源的弹性和可扩展性问题以及计算并行化问题
基于 surrogate	[41]	Odessa	阶段	性能	是	解决交互式感知应用的性能提升问题，该系统能够根据移动终端周围的环境自动地进行计算迁移和并行化决策
	[42]	AIOLOS	类	执行时间	是	基于 Android 平台的计算迁移中间件，支持自适应计算迁移，并基于 Android 为开发人员提供应用开发环境

3 典型的计算迁移系统

本节具体介绍和分析三个最具代表性的计算迁移系统，即 MAUI、Cloudlet 和 CloneCloud。介绍它们的系统架构以及在系统实现过程中采用的相关技术，并分析各系统的优缺点。三个系统分别体现了计算迁移三种不同的实现方法，即基于 surrogate、基于 cloudlet 和基于云。

3.1 MAUI

MAUI 是基于 surrogate 的计算迁移系统，其目标是支持细粒度（方法级）的计算迁移，最大程度地节省智能移动终端的能耗。

MAUI 系统架构如图 4 所示，在智能手机侧，MAUI 运行时包括三个组件：（1）Solver，负责提供调用决策引擎的接口；（2）Proxy，负责远程执行过程中的控制和数据传输；（3）Profiler，负责修改程序，收集程序的能耗、测量信息和数据传输需求。在服务器端，MAUI 包含四个组件，其中，profiler 和 proxy 提供和客户端对应组件相似的服务，solver 负责周期性地求解线性规划问题，controller 负责处理用户认证，并根据客户端请求分配资源来实例化

被划分好的应用。

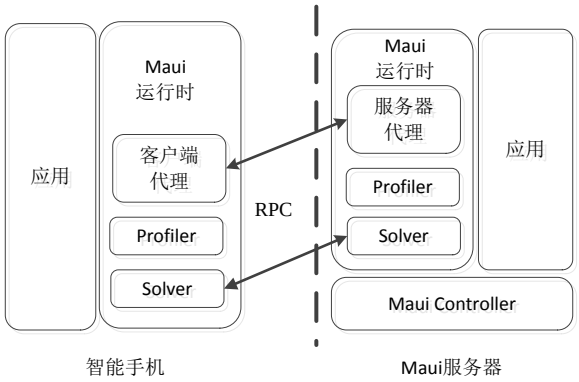

图 4 MAUI 架构图

MAUI 利用.NET 公共语言运行时的四个特性实现计算迁移：（1）使用代码可移植性产生两个版本的智能手机应用，一个版本运行在本地，另一个版本运行在远程服务器上。托管代码使得 MAUI 能够忽略移动设备和服务器不同的指令集架构；（2）使用程序反射特性自动识别被开发人员注释为”[Remoteable]”的方法，使用类型安全特性将应用状态发送给远程执行的方法；（3）对应用的每一个方法进行剖析（profile），并使用序列化特性确定其网络运输成本。

由于运用了.NET 特性，MAUI 体现出如下特点：（1）解决了系统的跨平台性，但仅限于对.NET 应用进行计算迁移；（2）需要应用开发人员预先对应用源代码中可远程执行的部分进行标注，因此难以对第三方软件进行划分，并且划分的自动化程度较低。

3.2 Cloudlet

文献[1]提出一种基于cloudlet的计算迁移方法，作者采用动态VM合成技术实现并验证了该方法。

动态 VM 合成过程如图 5 所示，步骤如下：（1）移动设备发送一个 VM 覆盖到已经运行着基础 VM 的 cloudlet；（2）cloudlet 基础设施把覆盖应用到基础 VM 上从而产生 launch VM，准备好为移动客户端提供服务；（3）移动设备向 launch VM 发送应用执行请求，并将应用设置为挂起状态；（4）launch VM 收到请求，从应用挂起的状态开始执行；（5）应用执行结束，将结果返回给移动设备，移动设备发起结束请求；（6）launch VM 产生 VM 残留，cloudlet 将 VM 残留发送给移动设备并丢弃 VM；（7）移动设备离开。

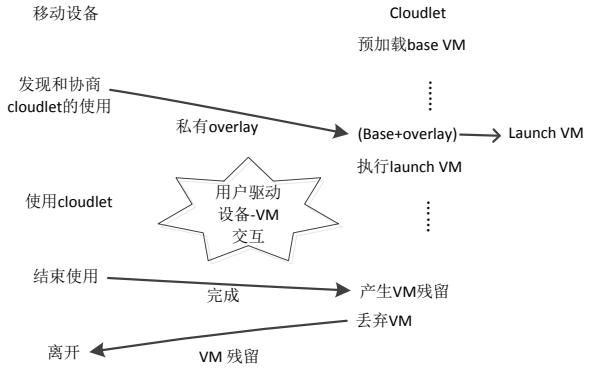

图 5 动态 VM 合成时间轴

文中的动态 VM 合成基于硬件 VM 技术，采用该方法实现的计算迁移系统具有如下特点：（1）对 cloudlet 基础设施的瞬态定制使得 cloudlet 能够自我管理和维护；（2）支持以 VM 为粒度的计算迁移，优点是这种计算迁移方法没有语言依赖性，缺点是动态 VM 合成的效率比较低；（3）将移动终端看作瘦客户端，所有计算都在 cloudlet 上执行，不能很好地利用移动终端的资源，也不适用于网络连接质量较差的环境；（4）能够对开发人员屏蔽底层平台的异构性。

3.3 CloneCloud

CloneCloud 是基于云的计算迁移系统，其系统架构如图 6 所示。其中，Profiler 负责为应用的每一次执行分别生成设备和云克隆端的执行成本模型——profile 树；划分分析器负责根据 profile 树，选出一系列需要远程执行的方法；迁移模块负责线程的挂起、状态打包、恢复和状态合并；管理器负责准备克隆镜像，并处理移动设备与云克隆之间的通信和同步问题。运行时负责决定使用哪一种应用划分方案。

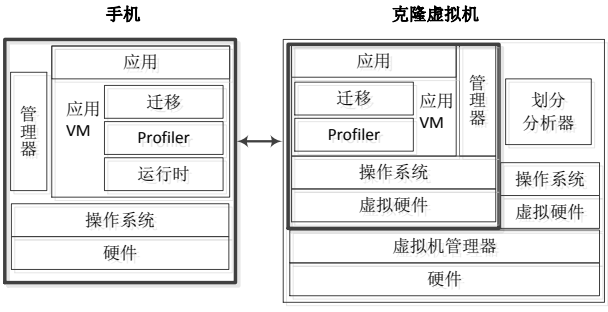

图 6 CloneCloud 系统架构

作者在 Android 平台上实现了 CloneCloud 系统原型，通过对 Dalvik VM 进行修改，来实现应用的动态剖析和迁移。CloneCloud 具有出如下特性：（1）良好的透明性。能够自动转换所有运行在应用 VM

上的移动应用，完全不需要开发人员的参与；（2）支持细粒度（即线程级）计算迁移；（3）良好的跨平台性。然而，CloneCloud 采用离线划分机制，即需要预先为各种执行条件（包括 CPU 速度、网络质量等）生成划分方案。由于不太可能预测和覆盖所有的执行条件，因此，和运行时划分系统相比，CloneCloud 的适应性相对较弱。

4 计算迁移系统的组成结构

第 3 节详细介绍和分析了三个经典的计算迁移系统，本节进一步从软件架构角度讨论一般计算迁移系统的内部组成结构、实现原理和使用到的相关技术。

我们提炼了计算迁移系统的功能模块，如图 7 所示。其中，应用划分模块、迁移执行模块、资源需求预测模块、资源监测与剖析模块是计算迁移系统的关键模块。

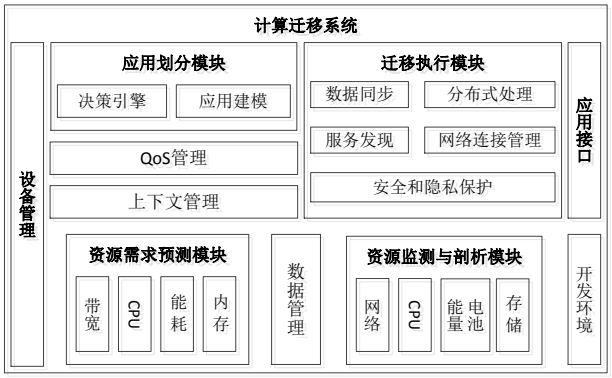

图 7 计算迁移系统功能模块

应用划分模块是计算迁移系统的核心模块，负责对应用行为建立模型，利用适当的算法生成候选划分方案，然后根据用户偏好、划分策略、移动终端上下文环境选出最优划分方案。

迁移执行模块负责为最优划分方案提供服务发现、网络连接管理、数据同步、分布式处理以及用户安全和隐私保护等执行机制，并且在应用运行过程中，收集和记录应用的资源使用情况。

资源需求预测模块负责根据过去的资源使用情况预测计算任务将来的资源使用需求（包括 CPU、内存、带宽和电池能量等）。

资源监测与剖析模块负责监测、记录可用资源的情况，并且为可用资源建立模型。这里的资源包括：移动终端和远程设备可用的 CPU 和内存、移动终端剩余电量和可用的网络带宽等。

下面分模块介绍计算迁移四个方面的内容（即应用划分、迁移执行、资源需求预测和资源监测与剖析）以及使用到的相关技术。

4.1 应用划分模块

4.1.1 应用划分的步骤

应用划分包含四个步骤：（1）为应用行为建立模型；（2）将应用划分问题转化为数学模型；（3）生成候选划分方案；（4）根据用户偏好选择最优划分方案。

（1）为应用行为建立模型。通常将应用行为抽象为一张成本图（成本指执行时间、能耗和内存使用量等）。目前存在三种方法获得成本图，第一种方法是在运行时对应用进行动态剖析[7]，第二种方法是对应用进行静态分析[56]，第三种方法则采用静态分析和动态剖析相结合[34]的方式。静态分析法简单，但是不能准确获取应用在运行时的成本信息，因此很难获得有意义的划分，动态剖析法虽然能够准确获得应用在运行时的成本信息，但是得到的成本图规模太大，将导致划分成本过高，因此，目前主要采取第三种方法获得成本图。

图 8 为应用成本图的示意图，其中，每个节点代表一个计算任务（方法、类、对象等），令 V 表示节点的集合，e_u、t_u 和 m_u 分别表示完成计算任务 u 需要消耗的能量、时间和内存，节点之间的边代表计算任务之间的调用关系，e_{uv} 和 t_{uv} 分别表示在节点 u 和节点 v 之间传输应用状态和执行结果时，所需要的通信能耗和时间。

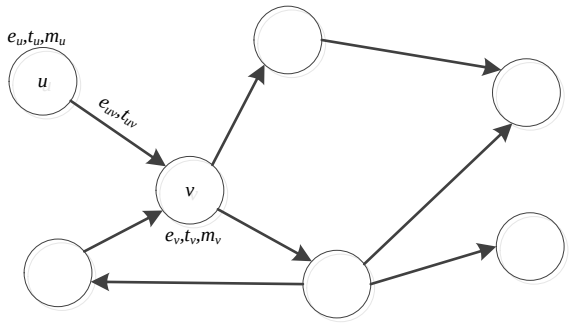

图 8 应用成本图示意图

（2）将应用划分问题转化为数学模型。通常将应用划分问题模型化为整数规划问题。作为示例，本文令 l_i 表示节点 i 的位置，$l_i \in (0,1)$，取值为 1 表示节点 i 被划分到本地，为 0 表示被划分到远程设备，从而将应用的划分问题转化为 0-1 整数规划问题。

在只考虑节能的情况下，当所有计算任务都在本地执行时，移动终端能耗可表示为：

$$E_l = \sum_{u \in V} e_u \tag{1}$$

应用被划分后，一部分计算任务在本地执行，另一部分计算任务在远程设备执行，此时，移动终端的能耗可表示为：

$$E_r = \sum_{u \in V} l_u \times e_u + \sum_{u,v \in V} |l_u - l_v| \times e_{uv} + e_{wait} \tag{2}$$

其中，e_{wait} 表示移动终端等待远程设备完成计算任务所需的能耗，包括 CPU 和无线接口空闲等待的能耗，于是，以节能为目标的最佳划分方案可以表示为：

$$\max E = E_l - E_r \tag{3}$$

$$s.t. E_l > E_r \tag{4}$$

在只考虑执行时间最短的情况下，当所有计算任务在本地执行时，移动终端的执行时间可表示为：

$$T_l = \sum_{u \in V} t_u \tag{5}$$

应用被划分后，移动终端的执行时间可表示为：

$$T_r = \sum_{u \in V} l_u \times t_u + \sum_{v \in V} (1 - l_v) \times t_v + \sum_{u,v \in V} |l_u - l_v| \times t_{uv} \tag{6}$$

于是，以执行时间最小为目标的最佳划分方案可表示为：

$$\max T = T_l - T_r \tag{7}$$

$$s.t. T_l > T_r \tag{8}$$

以移动终端内存使用量最小为目标的应用划分同理，在此不再赘述。

（3）生成候选划分方案。我们已经知道，图的最优划分是一个 NP 完全问题，解决办法通常是产生一系列备选划分方案，然后根据成本度量从备选方案中选出最优方案。常见的划分算法一般为基于 Stoer-Wagner 的 MINCUT 算法和分支-定界算法。但是，随着成本图中节点数量的增加，算法的复杂度呈指数型增长，因此在划分粒度较小的情况下，生成备选方案将产生巨大的计算和能耗成本，为了减小解的查找空间，通常采用如下方法：一种方法是使用修剪启发式算法来减小解的查找空间[15][16]，另一种方法是允许应用开发人员预先提供有意义的划分[9]或者为能够被远程执行的计算任务打上标记[7][35]，从而有效缩减备选方案的数量。

（4）选择最优划分方案。划分的最后一步是根据用户偏好，从划分备选方案中选出最优划分方案。用户偏好可以描述为不同度量指标的优先程度，包括节能优先、执行时间优先、内存使用量优先等。移动用户通常会同时关注多个目标，例如，希望在节能的同时能够获得较好的应用性能。一般采用效用函数来表达移动用户对不同度量指标的偏好次序，通过为不同的指标赋予不同的权重，来表达度量指标的相对重要程度。以用户同时关注能耗和执行时间为例，则某个候选方案的效用值可以表示为：

$$U = \omega_1 \times \frac{E}{E_l} + \omega_2 \times \frac{T}{T_l} \tag{9}$$

其中，ω_i 表示各个度量指标的权重，满足如下条件：

$$\sum_{i=1}^{n} \omega_i = 1, 0 \le \omega_i \le 1 \tag{10}$$

4.1.2 影响应用划分的因素

影响应用划分的因素如表 8 所示，将这些因素根据来源分为四类。

表 8 影响应用划分及决策的因素

来源	影响因素	影响范围
移动用户	行为模式、用户偏好	最佳应用划分方案的选择
开发人员	划分策略、划分触发机制	应用划分效率
移动设备	外部环境和内部环境	候选方案生成/能量优化效果
应用	输入、质量等级	候选方案的生成

第一类因素来自于移动用户，包括用户偏好和用户使用移动终端的行为模式。在不同场景下，用户偏好可能会发生变化，例如在电池电量不足的情况下，用户可能愿意牺牲一部分应用性能来换取更长的待机时间，而在电池电量充足的情况下，用户则希望获得最好的应用性能。除此之外，用户的行为模式也对最佳划分方案的选择有一定影响。文献[57]指出，移动设备能耗最终由用户行为决定。这意味着，即使针对相同设备和应用，以及相同的执行环境，由于用户不同，移动设备产生的能耗仍然会发生很大变化，这可能导致计算迁移系统产生完全不同的划分方案。因此，我们有必要理解用户的

行为模式，从而确定能量优化方案对用户体验的影响。然而，目前大多数计算迁移系统缺乏考虑用户行为对计算迁移的影响。

第二类影响因素来自开发人员，包括开发人员定义的划分策略、划分提示和划分触发机制。一些计算迁移系统为开发人员提供了接口，允许他们定义划分计划[9]，或标记出可以被远程执行的计算任务[7]，这使得系统能够减省自动识别可远程执行任务的时间，并减小划分方案的查找空间，极大提高了应用划分算法的效率。而应用划分触发机制，则能够有效避免系统在遇到任何资源变化时都对应用进行重新划分[15]；

第三类影响因素来自移动设备的上下文信息，包括移动设备的外部环境和内部环境。其中移动终端/服务器的 CPU 负载将影响移动终端的 CPU 能耗，无线网络接口类型（3G、Wi-Fi、WiMax 等）、网络质量和服务器缓存将影响无线接口能耗，从而间接影响候选方案生成结果，而移动终端可用内存大小和电量等因素将直接影响候选划分方案生成结果。此外，文献[7]指出，移动终端采用的节能管理技术会影响能量优化效果：如果无线网络时延较小，在传输应用状态时使用 PSM（power-save mode，节能模式）将导致应用总体能耗升高，只有当无线网络时延接近睡眠时间间隔（通常为 100ms）时，PSM 才能达到节能效果。

第四类影响因素和应用相关，包括应用的服务质量等级[47]、输入参数的内容和占用的存储空间[17]等，这些因素将影响应用的计算能耗和通信能耗，从而将导致计算迁移系统生成不同的候选方案。

4.1.3　划分考虑的问题

应用划分最终是要解决在什么时间将哪些计算任务迁移到何处执行的问题，下面从划分时机、划分粒度和计算任务的执行位置三个方面对应用划分问题进行描述。

（1）划分时机

根据应用划分的时机将划分方法分为静态划分和动态划分。静态划分指开发人员在软件设计和开发阶段，依据软件功能和逻辑将应用划分成多个部分。典型的 C/S 结构则采用了这种划分方法。然而，对移动应用而言，由于移动终端周围的环境（例如可用网络类型、带宽、网络时延等）可能不断发生变化，而且各种移动终端的硬件平台在配置上存在着较大差异，开发人员不可能为所有执行环境和硬件配置定制对应的应用划分方案，因此，该方法很难保证在所有环境条件下划分方案都是最优。

相比之下，动态划分发生在应用运行时，当移动终端可用资源不足或者周围环境发生变化时，计算迁移系统将触发应用划分。因此，动态划分体现出较好的灵活性，能够在一定程度上降低移动应用的开发难度，但另一方面，为了支持动态划分，计算迁移系统需要监测可用资源，并分析和预测应用的资源使用需求，这将为系统引入额外的存储和能量开销。目前几乎所有的计算迁移系统都采用动态划分方式。

（2）划分粒度

我们将已有应用划分粒度分为细粒度和粗粒度。细粒度包括方法[7][35][55]、类[15-16][19-20]、对象[12][44][46]和线程[34]级。粗粒度包括操作[10][11][13]、应用[4]和 VM[1]级。

细粒度的划分方法能最大程度地降低应用的能耗，但随着计算任务数量的增加，解空间呈指数增长，使用该方法将极大地降低最优解的查找效率，因此，它存在计算开销大、通信成本高和划分效率低等缺陷。为了弥补这些缺陷，在采用细粒度划分方法时，一般需要应用开发人员定义有意义的划分方案或标记出能够远程执行的计算任务，这意味着，该方法在某种程度上依赖于开发人员。因此，细粒度划分适用于规模较小的应用，此外，该方法还适用于网络质量变化较快的场景，因为即便远程服务器上某个计算任务执行失败，但由于粒度较小，将它回退到移动终端重新执行也不至于使用户体验受到太大的影响。

和细粒度划分方法相比，粗粒度的划分方法具有通信成本低、划分效率高等优势，因此能够支持自动应用划分，很大程度上减轻应用开发人员的负担。不足之处在于，迁移整个应用或VM需要较长的时间，当移动终端高速移动时，它有可能频繁地从一个surrogate覆盖区域进入另一个surrogate覆盖区域，这将导致短时期内应用或VM频繁地被迁移到不同的surrogate上，从而影响应用性能和用户体验。此外，在无线网络随机变化的场景下，执行较大的任务将增加由于网络质量差而导致任务执行失败的可能性。因此，粗粒度应用划分方法不适用于网络质量波动较大的场景。

（3）计算任务的执行位置

根据应用划分后计算任务的执行位置，将划分算法分为三类：第一类在移动终端和单个服务器上执行计算任务；第二类在多个移动终端上执行计算

任务[29-31]；第三类在移动终端和多台服务器上执行任务[35][44]。将计算任务分配到多个移动终端执行时，计算迁移系统的目标不再是为单个移动终端降低能耗，而是从全局角度出发，针对该移动设备集合降低能耗。将计算任务分配多台服务器上执行，则能够利用并行化执行提高应用的执行速度和可扩展性，这种执行方式对两类计算密集型任务非常有用：一类是递归程序或者能够使用分而治之的办法解决的问题，另一类是需要使用到大量数据的程序。

4.2 迁移执行模块

迁移执行模块负责执行应用划分模块产生的最优划分方案，为了支持应用的远程执行，该模块必须提供如下机制：

（1）服务发现机制

负责发现和定位移动终端周围环境中的计算资源，供移动终端根据需要进行选择。文献[1]使用Linux的Avahi机制支持服务浏览和发布，文献[16]使用UPnP和JINI来发现附近可以执行应用的surrogate，文献[29]通过整合Serverless Messaging（无服务消息）和XMPP（Extensible Messaging and Presence Protocol，可扩展通讯和表示协议）实现移动设备的发现和即时通信。文献[58]则提出了通用服务发现协议（versatile surrogate discovery service, VERSUDS），通过在已有的服务发现协议（JINI和Cooltown）之上增加虚拟层，为应用提供统一的API，从而避免当环境发生变化时，应用需要处理不同的服务发现协议。

（2）分布式执行机制

负责修改应用，增加远程执行需要的特性，并协调和控制任务的本地和远程执行。传统的分布式执行机制通常基于RPC [9][16]，新的分布式执行机制基于VM迁移技术[1][34]，该技术的优势是灵活，便于实现并行计算，目前存在的主要问题是VM迁移需要较长时间，如何提高VM迁移效率是值得进一步研究的问题。

（3）同步机制

负责同步移动终端和远程服务器的数据和执行状态。通常采用checkpoint技术来实现客户端和服务器端状态的同步。存在三种同步时机：周期性同步、根据需要进行同步或者选择时机进行同步[33]，举一个选择时机同步的例子：当移动终端发现高速Wi-Fi连接时，它可以进行更频繁的同步，从而避免使用低能效的3G连接。显然，同步频率将影响数据和执行状态的一致性，但是，过于频繁的同步将为系统引入额外的通信开销，因此，在设计计算迁移系统时，需要在同步频率和数据一致性之间进行权衡。

（4）回退机制

负责保证移动应用的可用性。在两种场景下，回退机制将发挥关键作用：一种在没有无线网络的场景中，回退机制能够确保应用在移动终端上仍然可用；另一种在移动应用运行时无线网络质量突然下降的场景中，回退机制能够确保应用能够立即回退到移动终端继续执行。

（5）安全机制

在计算迁移系统中，应用的一部分计算任务从移动设备迁移到远程服务器执行，该过程将涉及任务参数和用户信息的传递，以及由无线传输为用户引入的经济成本。因此，系统必须提供必要的安全机制以保证用户数据在传输过程中以及在服务器端的安全性、完整性和隐私性、应用执行结果的正确性，并保证用户的经济利益不受到损害。因此，计算迁移系统的安全机制主要涉及两个方面的内容：可信的运行环境、认证和安全通信。

可信的运行环境，主要指计算任务运行平台自身的安全性，它能够防止第三方恶意篡改应用代码和数据、或者干扰运行环境。从第2节描述的计算迁移系统的四种实现方法来看，基于云的计算迁移系统可以通过SLA向用户保证平台的服务质量（包括安全性），而其余三种方式，则难以保证为用户提供可信的运行环境。

认证和安全通信机制主要完成移动终端和远程服务器之间的身份认证以及在这两者之间建立安全的通信链接。文献[1]支持SASL（Simple Authentication and Security Layer，简单认证与安全层）框架，为应用提供可扩展接口，从而整合各种认证机制，并使用SSL（Secure Sockets Layer，安全套接层）协议在移动终端和cloudlet之间建立安全的TCP隧道，从而保证传输数据的安全性。文献[31]则设计了一个基于共享密钥和和会话密钥的协议，实现计算任务之间的相互认证，同时，作者认为，可以使用云平台提供的可信云计算服务，从而确保计算任务在一个安全的环境中执行。

4.3 资源需求预测模块

资源需求预测模块负责预测计算任务将来的资源使用需求，并根据预测结果计算出任务的本地执行成本和远程执行成本，然后，应用划分模块根

据成本信息进行划分决策。因此，预测信息的准确度将直接影响应用划分结果的正确性。资源预测模块必须具备两个特性：（1）能够准确地预测计算任务在本地和远程设备的执行成本；（2）由于资源预测模块运行在移动终端，因此，它使用的资源预测技术不能消耗过多的资源（即能耗、执行时间或存储容量）。

目前，大多数资源预测模块基于计算任务过去的资源使用情况，通过机器学习来预测将来的资源使用需求[9-11]。文献[9-11]采用统计的机器学习方法，对应用的资源使用日志数据进行拟合，从而产生一个拟合函数，该函数能够将输入参数映射到资源需求，在应用运行过程中，采用在线学习技术不断提高预测的准确性。该方法能够避免提前预测计算任务的资源需求，不足之处在于：（1）需要记录和存储应用的资源使用情况，引入了额外的文件读写和存储开销；（2）针对不同输入内容，应用实例的计算时间可能存在着较大差别，因此，初始资源需求预测模型的准确性将依赖于应用程序路径覆盖。文献[17]提出了 timeout 的概念，指能够从迁移中获益的计算时间，作者先计算出能够从迁移获益的最小执行时间，然后，使用统计信息不断优化最小执行时间，从而得到优化的 timeout。该方法的不足之处在于，当 timeout 结束时，如果计算任务还未完成，该任务将被迁移到远程服务器执行，那么，之前在移动终端消耗的能量就变成了额外开销。

4.4　资源监测与剖析模块

资源监测与剖析模块负责测量和记录可用资源的数量，在此基础上，建立模型，预测可用资源的将来的变化情况，包含两个方面功能：（1）监测和记录可用资源的数量，并根据应用划分的触发策略，在可用资源数量发生变化时，及时通知应用划分模块重新划分应用。可采用连续测量或周期性测量，测量周期越短，越能够及时触发应用划分，但测量的成本将增加；（2）基于可用资源的历史信息为可用资源建立模型，从而预测可用资源的数量，文献[18-19]提供了预测无线网络带宽的方法。

5　计算迁移系统的质量评价指标

目前，在如何评价计算迁移系统的质量方面还缺乏统一的标准。已有的计算迁移系统大多被用于概念验证，没有对其质量进行优化，有一小部分系统虽然考虑了质量优化问题，但主要关注如何提高系统的性能，缺乏对系统其他质量评价指标的考虑。本文总结和提炼计算迁移系统关键的质量评价指标，旨在为评估和优化系统质量提供基础和方向。

计算迁移是一种从软件层面解决移动终端资源受限问题的技术手段，计算迁移系统的质量评价应遵循通用的软件质量评估体系，但由于该系统应用于移动环境，所以对系统的自适应性、透明性、有效性以及安全性和隐私性要求更加严格。接下来从这四个方面讨论计算迁移系统的关键质量评价指标。

（1）自适应性

自适应性指计算迁移系统能够根据移动终端上下文环境，提供最优划分方案。自适应性对于计算迁移系统来说至关重要。移动环境最主要的特征是计算资源的不断变化性和无线网络的随机性。如果不论设备处于何种应用场景和资源环境，划分方案始终保持不变，则可能增加移动终端的能耗，降低应用的性能，甚至使应用无法正常运行。因此，计算迁移系统必须具备自适应性，才能提供满意的用户体验。

衡量自适应性的关键标准是灵敏度，指快速响应事件的能力。计算迁移系统的灵敏度研究的是系统如何及时地感知到可用资源的变化，并对这种变化做出准确而快速响应的问题。文献[15]将模糊控制理论运用在计算迁移系统中，从而使得系统能够根据资源变化情况及时触发应用的重新划分，文献[59]则利用控制系统领域测量动态响应的方法来测量和量化系统的灵敏度。

（2）透明性

透明性也称为不可见性，计算迁移系统的透明性意味着开发人员和用户完全感受不到计算迁移的存在。然而在实际中，不可能做到完全的透明，只能尽量避免干扰用户来近似地达到这个目标。

对开发人员而言，计算资源呈现出多样性，包括移动终端构成的云资源、cloudlet 以及各种商业云平台，此外，移动终端的硬件平台和操作系统也存在异构性，如果将计算资源和移动平台的异构性和多样性问题留给移动应用开发人员，将极大地增加应用开发难度，降低开发效率。因此，计算迁移系统有必要对开发人员屏蔽远程执行的复杂性和各类平台的异构性。

对移动用户而言，随着富移动应用的产生和流行，透明性变得越来越重要，因为它将直接影响到

用户的体验质量。这需要计算迁移系统具备获取用户意图、偏好和行为模式等能力，从而尽可能减少系统对用户的干扰。文献[22][35]采用效用函数来获取用户偏好。

（3）有效性

计算迁移系统的有效性是指通过使用计算迁移系统，用户能够达到获得预期的目标（例如节能、提升应用性能等），同时，不会因为使用该系统而付出超出预期的成本代价。这里的成本不仅指无线网络和远程计算服务本身引入的经济成本，还包括移动终端的存储空间、能耗等。有效性是计算迁移系统最基本的质量属性。

（4）安全性和隐私性

随着基于云的计算迁移方法的出现，用户数据的安全和隐私问题得到越来越多的关注。智能手机中存储着很多用户机密和隐私数据，例如，联系人、SIM 卡信息、信用卡信息、银行账户信息等，能否保证这些数据的安全和隐私，已经成为移动用户和企业用户在决定是否采用计算迁移技术时考虑的关键问题。因此，为了能够得到实际应用，计算迁移系统必须具备安全性和隐私性。

6 计算迁移系统参考架构

基于对计算迁移系统内部组成结构和关键质量属性的讨论，本节进一步尝试提出计算迁移系统的参考架构。

已有的计算迁移系统架构大多是两层架构，即系统运行在移动客户端和远程服务器端。其设计和实现基于三个前提：（1）已确定使用某一种计算迁移实现方法；（2）已选定某一个远程设备作为任务迁移的目标计算实体；（3）计算任务在执行过程中使用到的数据已经存储在选定的目标计算实体上。

然而，上述三个前提并不符合实际应用场景：（1）通过第 2 节对四种计算迁移实现方法的比较可以看出，它们有各自的使用场景、优势和不足，单一的计算迁移方法难以满足所有的应用场景；（2）在实际应用场景可能存在多个可供选择的远程计算实体；（3）计算任务在执行过程中使用到的数据可能并没有存储在目标计算实体上。

基于这些前提设计出来的架构存在如下几个问题：

（1）架构的自适应性较差。随着移动终端应用场景不断变化，采用单一计算迁移方法实现的系统将难以保证应用的性能和能耗；

（2）缺乏资源管理机制。这里的资源包括数据资源和计算资源。由于缺乏计算资源管理机制，当实际应用场景存在多个可供选择的计算实体时，计算迁移系统将无法解决多个计算实体的负载均衡问题；而缺乏数据管理机制，则会导致系统产生不必要的数据流动，从而影响应用的性能和能耗；

（3）缺乏对无线网络性能方面的考虑。在已有的计算迁移系统中，移动终端直接和远程计算实体通信，在实际应用中，随着客户规模的增大，如果仍然采用直接通信的方式，在服务发现阶段，有可能产生严重的广播风暴和服务发现延迟，从而影响计算迁移系统的性能。

针对已有计算迁移系统存在的问题，本文尝试提出一种新的基于三层架构的计算迁移参考架构，如图 9 所示。该架构由智能移动终端、计算节点和资源管理服务器组成。其中，计算节点包括智能移动终端、cloudlet 和远程服务器；资源管理服务器主要负责计算资源和数据资源的管理、计算节点的注册管理以及向智能移动终端反馈计算节点的相关信息。

在智能移动终端侧，应用代理主要负责根据资源管理服务器反馈的计算节点列表、用户偏好和智能移动终端上下文等信息确定合适的计算迁移实现方法，并从计算节点列表中选择最优的一个或多个计算实体作为目标计算节点；而决策引擎则提供接口，只负责发送划分请求和接收划分结果，应用划分结果由计算节点中的决策引擎生成。

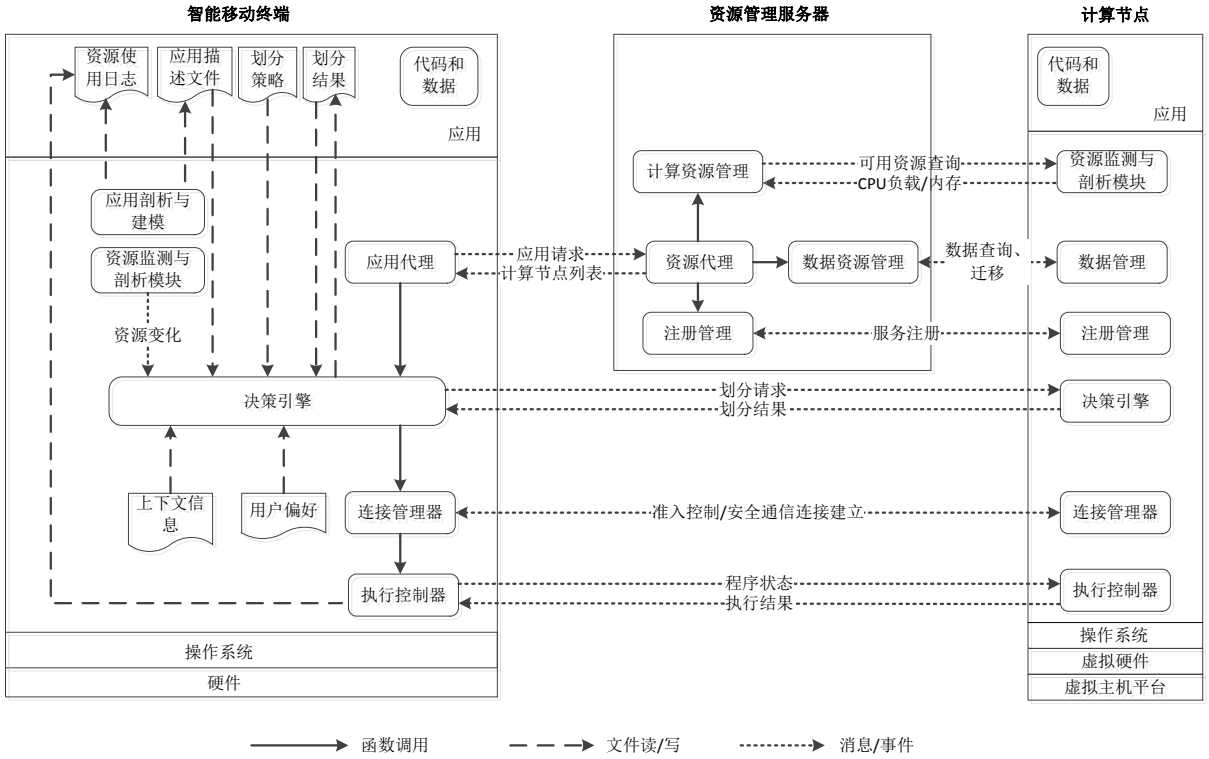

图 9 计算迁移系统参考架构

与已有的计算迁移系统架构相比，本文提出的参考架构具有以下优势：

（1）新的参考架构具有更好的自适应性。它不仅能够适应智能移动终端周围不断变化的网络环境，还能够整合不同的计算迁移实现方法，使得系统能够适应各类应用场景；

（2）新的参考架构能够彻底解决计算迁移的本质问题。计算迁移本质上是要解决在什么时间点将哪些计算任务迁移到哪里的问题。已有的计算迁移系统只解决了在什么时间点将哪些计算任务进行迁移的问题，而新的参考架构由于增加了资源管理功能，因此能够彻底解决计算迁移的本质问题；

（3）新的参考架构以数据为中心。已有的系统在计算任务的迁移过程中，没有考虑计算任务需要使用的数据不在目标计算节点的情况，这可能导致计算迁移过程中产生不必要的数据流动，从而降低计算迁移系统的性能。而新的参考架构以数据为中心，通过提供备份、预存储等机制以及相应的算法来避免或尽量减少数据流动，从而使得整个计算迁移系统的性能得以提升。

（4）新的参考架构能够更有效地利用资源。在新的架构中，增加了负载均衡机制，因此能够更有效地利用计算资源。此外，智能移动终端通过资源管理服务器来发现服务，有效地避免了移动终端和计算节点之间通过广播的方式交换服务通告和服务查询信息，因此能够更有效地利用无线网络资源。

7 总结和展望

计算迁移在分布式计算、普适计算和云计算阶段有不同的研究目的和内容。本文回顾计算迁移在不同阶段具有代表性的工作和进展，对已有研究成果进行归纳和比较，在分析三个典型的计算迁移系统的基础上，讨论一般计算迁移系统的内部组成结构和关键的质量评价指标，并试着提出计算迁移系统的参考架构。

云计算为计算迁移提供了更好的解决方案，也带来了新的研究内容，下面探讨计算迁移未来的研究需求：

（1）计算迁移技术的整合

目前，已经有各种各样的计算迁移系统采用不同方法实现计算迁移，有必要提供相关的机制，整合已有的方法，形成统一的计算迁移平台，从而使得计算迁移系统能够适用于不同场景，满足不同的用户需求。

（2）代码和移动数据的安全性

数据的安全性和隐私性是移动用户和企业用

户最为关心的问题之一。目前一些计算迁移系统实现了这两个特性，但在安全协议和加密算法的能效和延迟等方面还缺乏考虑。如何提高安全协议和加密算法的能效，从而减少移动终端使用计算迁移系统的开销，是值得进一步解决的问题。

（3）移动应用的开发

在移动云计算环境下，不论是把已有的应用转化成适合计算迁移的应用，还是开发和部署新的分布式应用，都更具挑战性，开发人员必须了解各种云平台的相关知识，从而增加了移动应用开发的难度和成本。如何对开发人员屏蔽各种云平台的异构性，简化移动应用的开发过程，是一项具有挑战性的工作。

（4）大数据环境下的计算迁移

在大数据环境下，移动终端的数据采集、处理、传输和访问将为移动用户引入不可预知的能耗、经济成本和时间成本。如何在移动终端和云之间分配数据处理和存储任务，从而优化移动终端的能耗、执行时间和经济成本，是尚待解决的问题。此外，传统计算迁移技术以计算为中心，为计算迁移系统增加数据感知特性是我们未来需要研究的内容。

参 考 文 献

[1] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, The case for VM-based cloudlets in mobile computing. IEEE Pervasive Computing, 2009, 8(4): 14-23.

[2] Othman M, Hailes S. Power conservation strategy for mobile computers using load sharing. Mobile Computing and Communications Review, 1998, 2(1): 44-50.

[3] Hunt G C, Scott M L. The Coign automatic distributed partitioning system//Proceedings of the 3rd USENIX Symposium on Operating Systems Design and Implementation. New Orleans, USA, 1999: 187-200.

[4] Rudenko A, Reiher P, Popek G J, et al. Saving portable computer battery power through remote process execution. Mobile Computing and Communications Review, 1998, 2(1): 19-26.

[5] Weiser M. The computer for the 21st century. Scientific American, 1991, 265(3): 94-104.

[6] Satyanarayanan M. Pervasive computing: vision and challenges. IEEE Personal Communications, 2001, 8(4): 10-17.

[7] Cuervo E, Balasubramanian A, Cho D, et al. MAUI: making smartphones last longer with code offload//Proceedings of the 8th International Conference on Mobile systems, applications, and services. San Francisco, USA, 2010: 49-62.

[8] Kistler, J.J., Satyanarayanan, M. Disconnected operation in the Coda file system. ACM Transactions on Computer Systems, 1992, 10(1): 3-25.

[9] Balan R K, Satyanarayanan M, Park S Y, et al. Tactics-based remote execution for mobile computing//Proceedings of the 1st International Conference on Mobile systems, applications and services. San Francisco, USA, 2003: 273-286.

[10] Flinn J, Narayanan D, Satyanarayanan M. Self-tuned remote execution for pervasive computing//Proceedings of the 8th Workshop on Hot Topics in Operating Systems. Schloss Elmau/Oberbayern, Germany, 2001: 61-66.

[11] Flinn J, Park S Y, Satyanarayanan M. Balancing performance, energy, and quality in pervasive computing//Proceedings of the 22nd International Conference on Distributed Computing Systems. Vienna, Austria, 2002: 217-226.

[12] Osman S, Subhraveti D, Su G, et al. The design and implementation of Zap: A system for migrating computing environments. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 361-376.

[13] Tilevich E, Smaragdakis Y. J-orchestra: automatic java application partitioning//Proceedings of the 16th European Conference on Object-Oriented Programming. Malaga, Spain, 2002: 178-204.

[14] Su Y Y, Flinn J. Slingshot: deploying 4stateful services in wireless hotspots//Proceedings of the 3rd International Conference on Mobile systems, applications, and services. Seattle, USA, 2005: 79-92.

[15] Gu X, Nahrstedt K, Messer A, et al. Adaptive offloading inference for delivering applications in pervasive computing environments//Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications. Fort Worth, USA, 2003: 107-114.

[16] Gu X, Nahrstedt K, Messer A, et al. Adaptive offloading for pervasive computing. IEEE Pervasive Computing, 2004, 3(3): 66-73.

[17] Xian C, Lu Y H, Li Z. Adaptive computation offloading for energy conservation on battery-powered systems//Proceedings of the 13th International Conference on Parallel and Distributed Systems. Hsinchu, Taiwan, China, 2007: 1-8.

[18] Wolski R, Gurun S, Krintz C, et al. Using bandwidth data to make computation offloading decisions//Proceedings of IEEE Internal Symposiums on Parallel and Distributed Processing. New York, USA 2008: 1-8.

[19] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. ACM SIGMETRICS Performance Evaluation Review, 2003, 30(4):41-49.

[20] Banerjee N, Rahmati A, Corner M D, et al. Users and batteries: Interactions and adaptive energy management in mobile systems//Proceedings of the 9th International Conference on Ubiquitous Computing. Zurich, Switzerland, 2007: 217-234.

[21] Rellermeyer J S, Alonso G, Roscoe T. R-OSGi: distributed applications through software modularization//Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference. Newport Beach, USA, 2007: 1-20.

[22] Balan R K, Gergle D, Satyanarayanan M, et al. Simplifying cyber foraging for mobile devices//Proceedings of the 5th International Conference on Mobile systems, applications and services. New York, USA, 2007: 272-285.

[23] Buyya R, Yeo C S, Venugopal S, et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 2009, 25(6): 599-616.

[24] Ha K, Pillai P, Richter W, et al. Just-in-time provisioning for cyber foraging//Proceedings of the 11th International Conference on Mobile systems, applications, and services. Taipei, Taiwan, China, 2013: 153-166.

[25] Verbelen T, Simoens P, De Turck F, et al. Cloudlets: bringing the cloud to the mobile user//Proceedings of the 3rd ACM workshop on Mobile cloud computing and services. Ghent, Belgium, 2012: 29-36.

[26] Verbelen T, Simoens P, De Turck F, et al. Adaptive application configuration and distribution in mobile cloudlet middleware. Mobile Wireless Middleware, Operating Systems, and Applications. Berlin Heidelberg: Springer, 2013.

[27] Lewis G A, Echeverría S, Simanta S, et al. Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments//Proceedings of the 36th International Conference on Software Engineering. New York, USA, 2014: 412-415.

[28]Verbelen T, Simoens P, De Turck F, et al. Adaptive deployment and configuration for mobile augmented reality in the cloudlet. Journal of Network and Computer Applications, 2014, 41(3): 206-216.

[29] Huerta-Canepa G, Lee D. A virtual cloud computing provider for mobile devices//Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services: Social Networks and Beyond. San Francisco, USA, 2010: 6:1-5.

[30] Shi C, Lakafosis V, Ammar M H, et al. Serendipity: enabling remote computing among intermittently connected mobile devices//Proceedings of the 13th ACM International Symposium on Mobile ad hoc networking and computing. New York, USA, 2012: 145-154.

[31] Kosta S, Perta V C, Stefa J, et al. Clone2clone (c2c): peer-to-peer networking of smartphones on the cloud//Proceedings of 5th USENIX Workshop on Hot topics in Cloud Computing. San Jose, USA, 2013: 1-5.

[32] Mtibaa A, Fahim A, Harras K A, et al. Towards resource sharing in mobile device clouds: Power balancing across mobile devices//Proceedings of the 2nd ACM SIGCOMM workshop on Mobile cloud computing. Hong Kong, China, 2013: 51-56.

[33] Chun B G, Maniatis P. Augmented smartphone applications through clone cloud execution//Proceedings of the 12th Workshop on Hot Topics in Operating Systems. Monte Verita, Switzerland, 2009: 9:8-11.

[34] Chun B G, Ihm S, Maniatis P, et al. Clonecloud: elastic execution between mobile device and cloud//Proceedings of the 6th ACM EuroSys conference on Computer systems. New York, USA, 2011: 301-314.

[35] KOSTA, S., AUCINAS, A., HUI, P., MORTIER, R., AND ZHANG, X. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading//Proceedings of the IEEE INFOCOM. Orlando, USA, 2012: 945-953

[36] Yang L, Cao J, Yuan Y, et al. A framework for partitioning and execution of data stream applications in mobile cloud computing//Proceeding of the IEEE 5th International Conference on Cloud computing. Hawaii, USA, 2012: 794-802.

[37] Shi C, Habak K, Pandurangan P, et al. COSMOS: computation offloading as a service for mobile devices//Proceedings of the 15th ACM International Symposium on Mobile ad hoc networking and computing. Philadelphia, USA, 2014: 287-296.

[38] Abolfazli S, Sanaei Z, Gani A, et al. Rich mobile applications: genesis, taxonomy, and open issues. Journal of Network and Computer Applications, 2014, 40(7): 345-362.

[39] Abolfazli S, Sanaei Z, Ahmed E, et al. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. IEEE Communications Surveys and Tutorials, 2013, 16(1): 337-368.

[40] Christensen J.H, Using RESTful web-services and cloud computing to create next generation mobile applications//Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications (OOPSLA). Orlando, USA, 2009: 627-634.

[41] Ra M R, Sheth A, Mummert L, et al. Odessa: enabling interactive perception applications on mobile devices//Proceedings of the 9th International Conference on Mobile systems, applications, and services. Washington, USA, 2011: 43-56.

[42] erbelen T, Simoens P, De Turck F, et al. AIOLOS: middleware for improving mobile application performance through cyber foraging. Journal of Systems and Software, 2012, 85(11): 2629-2639.

[43] Park S, Chen Q, Yeom, H Y. PIOS: a platform-independent offloading system for a mobile web environment//Proceeding of IEEE Consumer Communications and Networking Conference (CCNC). Las Vegas, USA, 2013: 137-142.

[44] Sinha K, Kulkarni M. Techniques for fine-grained, multi-site computation offloading//Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Newport Beach, USA, 2011: 184-194.

[45] Zhang Y, Liu H, Jiao L, et al. To offload or not to offload: an efficient code partition algorithm for mobile cloud computing//Proceedings of the 1st IEEE International Conference on Cloud Networking. Paris, France, 2012: 80-86.

[46] Niu J, Song W, Atiquzzaman M. Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. Journal of Network and Computer Applications, 2014, 37(1): 334-347.

[47] Zhang W, Wen Y, Wu D O. Energy-efficient scheduling policy for collaborative execution in mobile cloud computing//Proceedings of the 32nd IEEE International Conference on Computer Communications. Turin, Italy, 2013: 190-194.

[48] Verbelen T, Stevens T, De Turck F, et al. Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. Future Generation Computer Systems, 2013, 29(2): 451-459.

[49] Zhang L, Tiwana B, Qian Z, et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones//Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/software codesign and system synthesis. *Scottsdale*, USA, 2010: 105-114.

[50] Saarinen A, Siekkinen M, Xiao Y, et al. Smartdiet: offloading popular apps to save energy. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 297-298.

[51] Yoon C, Kim D, Jung W, et al. AppScope: application energy metering framework for android smartphone using kernel activity monitoring//Proceedings of USENIX Annual Technical Conference. Boston, USA, 2012: 387-400.

[52] Oliner A J, Iyer A P, Stoica I, et al. Carat: collaborative energy diagnosis for mobile devices//Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. Rome, Italy, 2013: 10:1-14

[53] Zhang X, Kunjithapatham A, Jeong S, et al. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Networks and Applications, 2011, 16(3): 270-284.

[54] Kemp R, Palmer N, Kielmann T, et al. Cuckoo: a computation offloading framework for smartphones. Mobile Computing, Applications, and Services. Berlin Heidelberg: Springer, 2012.

[55] Kristensen M D. Scavenger: transparent development of efficient cyber foraging applications//Proceedings of IEEE International Conference on Pervasive Computing and Communications. Mannheim, Germany, 2010: 217-226.

[56] Zhang Y, Huang G, Liu X, et al. Refactoring android java code for on-demand computation offloading//Proceedings of the 27th ACM SIGPLAN International Conference on Object Oriented Programming, Systems, Languages and Applications. Tucson, USA, 2012: 233-247.

[57] Shye A, Scholbrock B, Memik G. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures//Proceedings of the 42nd annual IEEE/ACM International Symposium on Microarchitecture. New York, USA, 2009: 168-178.

[58] Balan R, Flinn J, Satyanarayanan M, et al. The case for cyber foraging//Proceedings of the 10th ACM SIGOPS European Workshop. Saint-Emilion, France, 2002: 87-92.

[59] Noble, B.D., Satyanarayanan, M., Narayanan, D., Tilton, J.E.,Flinn, J., Walker, K.R. Agile application-aware adaptation for mobility//Proceedings of the 16th ACM Symposium on Operating Systems Principles. Saint-Malo, France, 1997: 276-287

附录X.

Zhang Wen-Li, born in 1979, Ph. D. candidate, lecturer. Her research interests include embedded real-time system and green computing.

Guo Bing, born in 1970, Ph. D., professor, Ph. D., supervisor. His current research interests include embedded real-time system and green computing.

Shen Yan, born in 1973, Ph. D., associate professor. Her research interests include distributed measurement systems, and robotics.

Wang Yi, born in 1976, Ph. D. candidate, lecturer. His research interest is embedded real-time system.

Xiong Wei, born in 1979, Ph. D. candidate, lecturer. His research interest is embedded real-time system.

Duan Lin-Tao, born in 1978, Ph. D., associate professor. His research interest is embedded real-time system.

Background

In recent years, resources (include computation, storage and energy) limitation has become increasingly distinct with the pervasive usage of the intelligent mobile terminals and the applications' ever-increasing requirement for resources. How to extend the resources of the mobile terminals has become the problem which needs to be solved urgently in the field of mobile computing. Computation offloading is an effective approach to extending the resources for mobile terminals. At present, various computation offloading systems are devoted into augmenting the mobile terminals. They address the problems of what parts of an application should be offloaded and what time these parts should be offloaded, while ignoring the issue of where these parts should be offloaded to. In addition, there is no unified architecture. This paper reviews the representative works on computation offloading in the context of three computation models (i.e. distributed computation, pervasive computation and cloud computation) and analyzes three classic computation offloading systems, then discusses some common problems such as the structure and quality attributes of the existing computation offloading systems. This work is partly supported by the State Key Program of National Natural Science Foundation of China under Grant No.61332001, the National Natural Science Foundation of China under Grant No. 61272104 and 61472050. Our group has been working on the energy consumption optimization for embedded system and computer networks. Many papers have been published in international conferences and journals, such as ICESS, GreenComm, Journal of Software, Chinese Journal of Computers and Journal of Systems Architecture. This paper tries to present the reference architecture, which solves the problem of where the remote parts of an application should be offloaded to through resource manager and data-centric computation offloading algorithm. It also proposes the key quality evaluation metrics for the computation offloading systems.