计算机系统体系结构的层次设计

林 闯 薛 超 胡 杰 李文焯
(清华大学计算机科学与技术系 北京 100084)

摘 要 计算机系统由许多连通的层和子系统组成,这些层和子系统的交互模式复杂,整体体系结构设计尤为重要.在计算机系统体系结构演变的过程中形成了一些基本设计原理,其中层次设计是设计大规模系统体系结构的有效途径.从 Dijkstra 的工作开始,计算机系统体系结构的层次设计原理已经被提出很长时间,是计算机系统体系结构设计的重要组成部分.它广泛存在于计算机体系结构设计、网络体系结构设计、云计算、网络虚拟化、软件工程以及计算机科学的很多其他分支.计算机科学技术的演变和革新异常频繁,适用范围广的层次设计模型框架和层次设计方法尤为重要.虽然有不少工作对体系结构层次设计进行研究,但很少有工作对层次设计原则和方法的内涵进行探索,同时缺少统一的层次设计模型框架和评价指标.现有工作的不足主要表现为:(1)对计算机系统层次设计的描述通常是非形式化阐述;(2)现有的层次设计分析以具体系统和应用分析为主,缺少对层次设计机制内涵的理解分析;(3)现有层次设计模型主要局限于所研究的对象系统,缺少统一的层次设计模型框架和评价指标.针对计算机系统层次结构设计的上述不足,该文首先给出了层次设计相关的基本概念及其形式化定义,然后对层次设计研究现状从层次模型设计、层次构件设计、层次跨层设计和层次覆盖设计四个方面进行归类综述.层次模型设计主要包含层次描述模型和层次量化模型,对层次结构针对对象系统特征进行数学描述和推导;层次构件设计将层次结构的某一子结构以单一或较少模块抽象来实现;层次跨层设计是指打破既定层次结构,根据特定需求生成新的层间交互关系;层次覆盖主要是指以虚拟节点和逻辑连接构成的灵活的虚拟平面设计.在此基础上,对层次设计的内涵、设计原则、主要机制和设计路径进行探究和归纳.该文认为简化和效率是计算机系统体系结构层次设计的两个设计原则,抽象和虚拟是支撑设计原则的两个设计机制.相应地,该文给出计算机系统体系结构层次设计复杂性和性能的模型评价框架,得到一些基本定理.该文还对超级计算机系统、软件定义网络和云计算三个层次设计经典系统例子进行讨论,并在文章的结尾对计算机系统体系结构层次设计的进一步研究进行展望.

关键词 计算机系统;体系结构;层次设计;模型评价;抽象;虚拟;复杂性;性能
中图法分类号 TP302 **DOI 号** 10.11897/SP.J.1016.2017.01996

Hierarchical Architecture Design of Computer System

LIN Chuang XUE Chao HU Jie LI Wen-Zhuo
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract Computer system is composed by several layers and sub-systems which link with each other with complex interactions between them. The overall architecture design of computer system is very important. Many design principles were proposed during the evolution and development of computer system architecture, among which hierarchical architecture design is an efficient way for large system design. Since the literature elaborated by Dijkstra, hierarchical architecture design principle of computer system architecture has been proposed for many years, and it is an important part of computer system architecture design. Hierarchy design exists widely in computer science and system, such as computer architecture design, network architecture design, cloud computing,

收稿日期:2016-07-13;在线出版日期:2017-03-22.本课题得到国家自然科学基金(61472199)和清华大学自主科研项目(20121087999)资助.**林 闯**,男,1948 年生,博士,教授,博士生导师,主要研究领域为计算机网络、系统性能评价、网络安全分析和 Petri 网.E-mail:chlin@tsinghua.edu.cn.**薛 超**,男,1988 年生,博士研究生,主要研究方向为计算机网络体系结构的性能评价与优化、云计算虚拟资源调度.**胡 杰**,男,1990 年生,博士研究生,主要研究方向为软件定义网络的性能评价与优化.**李文焯**,男,1991 年生,博士研究生,主要研究方向为大数据资源的管理调度与优化.

network virtualization, software engineering and many other branches. Computer science and technology is evolving and renovating with a high frequency, and it is extremely important to propose a general model framework and design mechanism for hierarchy design. Though some literatures have been explored on hierarchy design of system architecture, few works explored the essence of the principles and the methods of hierarchy design, let alone general model framework and evaluation metric. The main insufficiencies appear as follows: (1) the descriptions of hierarchy design lack of formal descriptions; (2) the analyses of hierarchy design mainly focus on specific system and application lacking of understanding on the essence of the principles and the methods; (3) the models of hierarchy design are also limited to the target system lack of general model framework and evaluation metrics. This paper shed lights on these mentioned issues of hierarchical architecture design of computer systems in the following procedures. Firstly, the basic concepts and preliminary formal definitions related to hierarchy design are given. After that, this paper surveys the state of the arts of hierarchy design through four categories named as hierarchy model design, hierarchy component design, hierarchy cross-layer design, and hierarchy overlay design respectively. Hierarchy model design includes description model and quantification model giving mathematical description and deduction to specific characteristics of the target system; Hierarchy component design abstractsa given substructure of the target system into a simple one with its core function; Hierarchy cross-layer design generates new virtual links and interactions beside the original hierarchical structure for specific demands; Hierarchy overlay design uses virtual nodes and logical links to form a flexible virtual plane. On that basis, the essence of basic concepts, design principles, main mechanisms, and implementation method of hierarchy design are explored. We summarize simplification and efficiency as two essential principles. They also imply the objectives of hierarchy design at the same time. These two principles are achieved by abstraction mechanism and virtualization mechanism which are formally defined by plane mappings. Accordingly, we establish an evaluation framework for complexity and performance of hierarchical structure, give their formal expressions, and obtain some theorems. This paper also makes essential introduction and discussion on the hierarchical structure of three typical systems, i. e. super computer system, software defined network, and cloud computing system with complexity and performance evaluation methodology. Conclusion and prospective future research challenges are summarized at the end of this paper.

Keywords　computer system; architecture; hierarchy design; model evaluation; abstraction; virtualization; complexity; performance

1　引　言

计算机系统属于工程系统，由许多连通的层和子系统组成，这些层和子系统之间的交互模式复杂，并且可能随时间变化. 从第一台通用电子计算机问世以来，计算机系统在性能、存储等方面飞速发展. 这既得益于计算机系统生产技术的发展，也得益于计算机系统体系结构的优化设计和创新[1]. 系统体系结构是系统的全局视图和主要结构，包括系统的组成和交互特征，贯穿包括前期功能区分设计等在内的系统设计实现的所有阶段，综合考虑系统的功能需求、属性指标和约束条件等许多方面[2]. 与具体细节设计、实现不同，体系结构或者说体系结构设计是一个更高级别的抽象，侧重各个组成部分的外在可见属性[3]. 对系统体系结构的深入研究可以加深对对象系统的理解，有助于系统更好地设计和实现[4]，缺少理论基础、不成熟的体系结构会影响部分乃至整个系统的运转[5-6]. 传统的计算机体系结构指计算机的概念性结构与功能特性，包括指令集体系结构(Instruction Set Architecture, ISA)、组成(Organization)和硬件(Hardware)，负责在不同的层次分配软硬件功能和确定软硬件界面.

在计算机系统体系结构演变的过程中形成了许

多基本设计原理,比如通过时间重叠、资源重复及资源共享充分利用并行性,局域性原理[7]和 Amdahl 定律[8].另外,一些研究对体系结构的内涵和表示方面进行探索,软件体系结构设计还形成了体系结构描述语言(Architecture Description Language,ADL)和模块互联语言(Module Interconnection Language,MIL)等描述技术. Harrison 等人[2]和 Eden 等人[9]对体系结构设计过程中相关概念的内涵进行区分,如体系结构、设计、实现,体系结构设计模式、设计策略、指标属性等. Tang 等人基于文献[3,10-12]将体系结构设计原理归纳为 9 个方面[13]:设计约束、假定、优点、缺点、开销、复杂性、结果、实现确定性和多设计方案折中机制,并且对这些设计原则在软件工业界的影响进行了调研.这些工作都对计算机系统体系结构设计的进一步发展起到了帮助作用.

层次设计是设计大规模控制系统体系结构的有效途径[14],是计算机系统体系结构设计的重要组成部分,在计算机领域各种系统体系结构设计中有着广泛应用.从计算机语言角度,可以把传统的计算机系统按照功能划分为从应用语言虚拟机级向下到微程序机器级的多级层次结构,高级语言以低级语言为基础,功能更强,对用户和开发人员更加友好. Dijkstra 将层次模型应用在计算机操作系统的设计中,将操作系统分为调度层、分页层、终端与操作系统通信层、I/O 管理层、用户程序层、用户层,上层的设计仅仅依赖于相邻的下层[15].开放系统互连(Open System Interconnect,OSI)参考模型将层次模型运用于网络系统互联设计中,形成包括应用层、表示层、会话层、传输层、网络层、数据链路层、物理层的七层协议模型,每一层可以独立演化,简化网络互联问题复杂性[16].更多地,如面向服务架构(Service Oriented Architecture,SOA)[17]、云计算(Cloud Computing)[18]、软件定义网络(Software Defined Network,SDN)[19]、物联网[20]、智能交通系统[21]、片上网络(Network on a Chip,NoC)[22]、分层路由[23]、蜂窝网络资源调度[24]、电力网控制[25-26]等都有基于层次设计的体系结构研究和实现.层次结构还存在于互联网的命名结构[27]、量子计算[28]和生物系统[29-30]中.

计算机科学技术的演变和革新异常频繁,适用范围广的层次模型框架和层次设计方法尤为重要.虽然层次结构设计可以应用到计算机系统的各个方面,并且有很多工作对体系结构层次设计进行分析研究,但是这方面的研究和探讨仍需进一步加强,主要表现为:(1)对计算机系统层次设计的描述通常是非形式化阐述;(2)现有的层次设计分析以具体系统和应用分析为主,缺少对层次设计机制内涵的理解分析;(3)现有层次设计模型主要局限于所研究的对象系统,缺少统一的层次设计模型框架和评价指标.我们以上述不足为切入点,给出了计算机系统体系结构层次设计涉及基本概念的形式化定义,并对相关研究现状进行综述.在此基础上,尝试对计算机系统体系结构层次设计的基本原则、主要机制的本质进行理解,同时给出了一个复杂性和性能指标评价框架,得到了一些有益结论.最后对三个计算机系统体系结构层次设计典型例子进行了讨论.

本文第 2 节对计算机系统体系结构层次设计所涉及到的基本概念的语义和模型框架进行形式化定义和介绍,并给出复杂性和性能评价指标;第 3 节从层次模型设计、层次构件设计、层次跨层设计、层次覆盖设计四个方面对计算机系统体系结构层次设计研究现状进行综述介绍;第 4 节对计算机系统体系结构层次设计的设计原则、设计机制和设计路径的语义进行介绍,并给出相应形式化表达;第 5 节以我们的模型框架为基础,给出了计算机系统体系结构层次设计的复杂性和性能评价方法,并给出了层次设计机制对复杂性影响的一些形式化结论;第 6 节对三个计算机系统体系结构层次设计经典系统例子——超级计算机、SDN 和云计算进行介绍讨论;在第 7 节对全文进行总结并对可能的未来工作进行探讨.

2 基本概念和定义

在计算机系统体系结构发展的过程中,新技术的产生和演变通常是较快的,但是层次模型[31]及其设计方法的基本理念的发展和演进则是非常缓慢的.计算机系统体系结构是一个整体,我们在研究其层次设计时,会依据研究的范围将体系结构整体分解为多个组成部分,不同的范围就确定了分解后不同部分概念、定义和功能的差异.结合现有的层次设计相关工作,我们将这些具有差异的概念归类为五个基本概念:资源、模块、接口、层、层次,分别放置于系统架构的物理平面(Physical Plane)和逻辑平面(Logical Plane)中.每一个基本概念都可以映射到计算机体系结构中的物理或功能实体,并且它们之间也不相互独立,如图 1 给出了五个基本概念以及两个平面的映射对应关系.

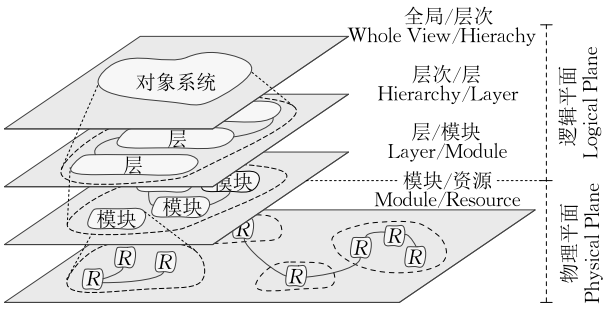

图 1　体系结构层次设计模型框架

这些概念已经存在于现有的一些技术和设计理念中，我们从层次设计的角度对它们进行理解和表达，以便更加深入地对层次模型设计进行理解. 下面我们对这些基本概念分别进行阐述. 平面 P、结构 G、路径 π 和连通分支 CC 四个基础概念的定义见附录.

2.1　资源、模块和接口

Perry 等人[32]认为体系结构由组成部件、组织形式和基本原理组成. 在我们对体系结构层次设计的讨论中，我们将前两者统称为资源，其中组成部件为物理资源(Physical Resource)，组织形式和功能配置为虚拟资源(Virtual Resource). 具体来讲，我们沿用 Foster 等人[33]的表述，计算机系统体系结构设计需要用到的资源包括计算资源、存储资源、网络资源、代码资源、目录索引资源. 计算、存储和网络资源为物理资源，代码资源、目录索引资源为虚拟资源，它们均位于物理平面，是计算机系统体系结构层次设计的基础. 于是我们用集合的方式定义资源.

定义 1(资源，Resource).　资源为一个集合 $R=Physical \cup Virtual$，包括物理资源和虚拟资源.

模块的概念定义在资源概念的基础上. 模块位于逻辑平面，是基本、功能完整、可重用的对象，可以是一个或一组资源，是计算机系统体系结构层次设计基本单元，通常只关注模块外在功能和交互特征，不需要了解模块的内部结构. 有些模块通过标准被严格定义，有些模块则可以动态灵活设置. 模块化(Modularization)是与模块概念相伴而生的概念，在计算机科学中通常也被称为“分治法(Divide and Conquer)”，它将系统划分为一组相互作用的子系统，建立子系统之间交互逻辑和界限. 模块化通常要使子系统具有相对的独立性，尽量简化子系统之间的相互作用，从而降低系统的规模复杂性.

定义 2(模块，Module).　模块由资源集合以及这个资源集合的结构构成，$m=(R_m, G(R_m))$. 特别地，$m^{-1}=R_m$. 一般不考虑模块的内部结构，$m=\Phi$ 称作空模块.

接口定义了一个给定模块的输入和输出规则，并隐含着输入和输出的关系，即模块在该接口下的功能，它将模块看成一个黑盒，并不关心模块内部的具体结构. 通常情况下，接口的设计都需要保证可以处理符合输入结构的任意的输入情况，最终给出在一定范围限定之内符合逻辑的输出，并且符合与之交互的模块接口的输入结构. 在外界看来，与之进行交互的接口就是模块本身，是模块逻辑的延伸，没有接口的模块是一个孤立系统.

定义 3(接口，Interface).　接口是模块的逻辑延伸，由二元组构成 $i_{m_x,m_y}=(m_x, m_y)$，$m_x, m_y \in M$，其中 m_x 表示(可能存在的)请求输入模块，m_y 表示对象模块.

连接是不同模块通过相应接口互联从而可以进行交互的一种有向关系，通常具有请求和应答的关系，而一组模块对象及其之间的连接关系称为一个结构. 在接口定义的基础上，定义连接如下.

定义 4(连接，Connection).　连接是接口的实例，由二元组构成 $c_{m_x,m_y}=(m_x, m_y)$，其中 $m_x, m_y \in M$. 即表示模块 m_x 与模块 m_y 的一个连接，是 m_y 接口的一个实例. 连接与所在平面结构的连接边一一对应. 连接的端点是服务访问点(Service Access Point，SAP)，即在同一系统实体之间进行信息交换的接口.

2.2　层和层次

层是一个或一组功能、规模或其他方面类似模块的集合，集合内部的模块之间可以交互，对层外开放交互的模块的接口也称作该层的接口. 层的结构并不一定是连通的，可能存在多个互不相连的数据层独立结构和控制层联通，但是其本身并不连通，这种形式的层是由功能类似的并行结构组成的.

定义 5(层，Layer).　层由一个模块集合以及这个模块集合的内部结构构成：$l=(M_l, G(M_l))$，并且将存在连接 $c_{m_x,m_y \mid m_x \in M, m_x \notin M_l, m_y \in M_l}$ 的接口 i_{m_x,m_y} 统称该层的接口 i_l.

层次是一个或一组层以及不同层之间交互模式的集合，该集合包括对象系统的所有组成部分，并且规定了集合中不同元素之间是否有连接，如何连接. 需要注意的是，一个层次的结构一定是连通的，并且传统的层次设计结构中，层次中的层只与相邻层通过连接进行交互. 与层和模块的关系类似，可以得到层次的概念.

定义 6(层次,Hierarchy). 层次由层集合以及这个层集合的内部结构构成:$h=(L_h,G(L_h))$.

层和层次的概念是具有相似性的,区别在于对已知系统划分层次、层的不同动机和观察方法.具体来讲,层规定了集合中模块之间的连接规则,但是它并不强调这些连接规则,甚至允许没有连接,将该集合看作一个整体,按照一个更大的模块来理解.与层不同,因为层次就是研究对象系统的体系结构,层次在规定了集合中不同层之间的连通性及连接规则的同时,强调这些连接逻辑,而弱化将层次集合看作整体.

综上所述,从模块到层,从层到层次都是在应用集合定义和图结构对系统物理平面和逻辑平面的概念进行抽象和组织,是对现有系统从不同角度进行的观察,并不对现有系统增加新的功能.

2.3 复杂性

从 McCabe[34] 开始,对于系统体系结构复杂性的研究经历了非常长的时间,许多工作给出了不同的理解.现有对系统体系结构复杂性评价指标的研究普遍存在需要人为主观评价的参数.Efatmaneshnik 等人[35]将系统的结构看作一个连通图,在此基础上将系统复杂性中的客观复杂性部分和主观复杂性部分分离,可以得到如下所示系统复杂性的综合表达式:

$$K(S)=\mathcal{F}(\mu(S),D(S,R)) \tag{1}$$

其中,K 指系统总复杂性,是系统客观复杂性 $\mu(S)$ 和主观复杂性 $D(S,R)$ 的函数.主观复杂性由对象系统 S 与参考系统 R 结构拓扑距离来定义,通常是基于最大公共子图(Maximum Common Subgraph,MCS)的形式化表达式[35-37]. $\mathcal{F}:Z^2\rightarrow Z$ 是关于两个变量的单调递增函数.从对系统的维护成本的复杂程度来考虑,客观复杂性需要满足如下准则,即原系统(结构)的客观复杂性严格大于子系统(结构)的客观复杂性.

基于这种复杂性结构,我们给出了系统层次结构复杂性评价指标的定义.

定义 7(复杂性,Complexity). 记 P 为对象系统的一个平面,v 为该平面实体,$CC(P)=\{cc\}$ 为 P 平面结构的连通分支集合,则该平面的复杂性定义如下:

$$K(P)=\sum_{cc\in CC(G(P))}[K'(cc)+\sum_{v\in cc}K(v^{-1})] \tag{2}$$

v^{-1} 为该平面下一平面映射到 v 的实体结构,对于最底层平面的实体 $v^{-1}=\phi$, $K(\phi)=0$; $K'(cc)$ 为该连通分支复杂性,即图的结构复杂性,我们采用如下定义:

$$K'(cc)=\mu(cc)\times[1+D(cc,R_{cc})] \tag{3}$$

对于主观复杂性 $D(S,R)$,定义如下:

$$D(S,R)=1-\frac{\mu(MCS(S,R))}{\mu(S)+\mu(R)-\mu(MCS(S,R))} \tag{4}$$

其中,$MCS(S,R)$ 为 S 与 R 最大公共子图.

式(3)中主观复杂性与客观复杂性进行乘积,是将多目标评价转化为单目标评价的过程.更一般地,可以给 $\mu(cc)$ 添加幂次系数 α 将两个目标进行加权乘积转换为单目标,α 为权重系数.在这种情况下,对于不同系统的设计和需求,主观复杂性的倾向性可能不同,权重系数 α 会有差异.根据对象系统的评价实际情况不同,同位于对象系统"最底层"平面上节点的复杂性也有可能是不同的.

2.4 性 能

性能评价是计算机系统层次结构评价的重要方面.计算机网络和计算机系统的性能包括多个方面的多种指标,如表征计算机系统能够正常工作的可靠性、可用性等,表征计算机系统处理能力和效率的吞吐率、响应时间、利用率等.在一些工作中将多种评价指标用效用(Utility)统一进行表达.效用是计算机系统体系结构设计需要考虑的重要评价指标,反映系统在特定体系结构设计下的运行效果,在不同的具体系统中有着不同的定义,可以具体为多种属性和对象的目标函数.在对效用进行具体量化表达的基础上,对体系结构针对效用进行优化设计的问题被称为效用最大化问题(Utility Maximization,UM),一般是一个单变量优化问题.Tang 等人[12]给出了一种系统体系结构效用的具体量化表达.他们同时考虑定性原理(Qualitative Rationale,QuR)和定量原理(Quantitative Rationale,QaR),分别计算体系结构设计不同方案的收益和开销,给出体系结构设计期望回报(Expected Return Ratio,ER)的表达式:

$$ER=\frac{EB}{EC}=\frac{(1-OCR)\times ABI}{(1+ICR)\times ACI} \tag{5}$$

其中,$EB=(1-OCR)\times ABI$ 为收益函数,由结果确定性(Outcome Certainty Risk,OCR)和体系结构收益指数(Architecture Benefits Index,ABI)来计算,$EC=(1+ICR)\times ACI$ 为开销函数,由实现确定性(Implementation Certainty Risk,ICR)和体系结构开销指数(Architecture Cost Index,ACI)来计算.Kelly 等人[38]以效用为目标对网络体系结构进行评价,给出了网络效用最大化(Network Utility

Maximization，NUM)的一般表达

$$\begin{aligned}&\text{maximize}\sum_s U_s(x_s)\\&\text{subject to } \boldsymbol{Rx}\leq \boldsymbol{c}\end{aligned}\tag{6}$$

其中，$U_s(x_s)$是服务 s 的效用函数，$\boldsymbol{R}$ 为路由矩阵；x_s为服务 s 获得的带宽，$\boldsymbol{x}$ 为带宽向量，$\boldsymbol{c}$ 为链路容量向量，最终用$\sum_s U_s(x_s)$来表达对象系统效用.

基于上述讨论，我们给出系统层次结构性能评价效用评价指标的定义.

定义 8(效用，Utility).　记 P 为对象系统的一个平面，v 为该平面实体，$CC(P)=\{cc\}$为 P 平面结构的连通分支集合，则该平面的效用定义如下：

$$U(P)=\mathcal{F}(U'(G(P)),U(V^{-1}))\tag{7}$$

其中，$U(P)=U(G(P))$与$U'(G(P))$为结构效用函数，$U(V^{-1})$为该平面下一平面映射到该平面节点集$V=\{v\}$的实体结构，与复杂性定义类似，$U'(G(P))$只与结构有关. 函数$\mathcal{F}$的表达形式具有多种表达评价方式，可以是如式(6)所示多个独立效用函数的加和，也可以是多目标优化意义下多个效用函数的 Pareto 最优，或是多个对象效用函数之间的博弈优化表达式.

3　层次设计研究现状

计算机系统体系结构层次设计的方式是多种多样的，不同的资源、功能可以部署在不同的层和模块中，某些设计方案在特定方面会比其他设计更优. 它的发展和更新是一个十分缓慢的过程，一个概念、技术和结构的出现、演进需要很长时间，在演化过程中，也会受到技术或非技术因素影响. 在不同应用环境中，形成了多种体系结构层次设计概念、技术和结构，我们将从四个方面对其进行介绍：层次模型设计、层次构件设计、层次跨层设计和层次覆盖设计.

3.1　层次模型设计

层次模型设计首先要针对对象系统特征，匹配合适的数学模型进行描述、建模；然后对数学模型依据对象系统实际进行条件约束；最后对数学模型进行推导，得到对对象系统有帮助的性质和结论. 层次模型的相关工作主要以应用为主，有层次描述模型和层次量化模型两种.

层次描述模型是层次结构模型的直观表达，是层次模型进行语义分析和验证的基础. Taylor 等人[39]给出分布式系统的三层层次描述模型，将系统分为前端层、中间层和后端层，分别具有不同的特征，并且相邻层之间存在请求和应答. Zave 等人[40]给出网络层次结构的层次描述模型，定义了层、层内部的组件以及各个组件之间的功能、算法和协议，再用层构成整体结构需求来分析网络的移动性. 概念属性研究方面，主要分为定量研究和定性研究. Marmsoler[41]介绍了体系结构设计的形式化方法，并且基于此对软件体系结构的层次设计建立层次描述模型，给出层和层次结构配置的形式化概念，并进行了语义和语法分析. 他们从服务的视角引入端口集合 PORT 和服务集合 SERVICE，其中 PORT 又细分为互不相交的入端口集合$\mathcal{I}$和出端口集合$\mathcal{O}$. 端口集合和服务集合的关系由映射 $type$ 表示，这里$\wp$指幂集. 在此基础上定义层 $layer$ 和层次结构，即配置 $config$ 如下，并指出约束条件.

$$\begin{aligned}&type:\ \text{PORT}\to\wp(\text{SERVICE})\\&layer=(I,O,f),I\subseteq\mathcal{I},O\subseteq\mathcal{O},f:\bar{I}\to\wp(\bar{O})\\&config=(L,A),L\subseteq\mathcal{L},A\in(L.in\mapsto L.out)\end{aligned}\tag{8}$$

其中 $L.in\mapsto L.out$ 中一个入端口只能映射到一个出端口. 他们的工作基于服务抽象，可以描述多种类型的服务.

层次量化模型与描述模型相比更加细致，以对象系统特定属性为基础进行建模分析、资源配置和优化控制. 图 2 给出了体系结构层次设计量化模型框架. 分层即优化分解(Layering As Optimization Decomposition，LAOD)[42-43]是一个具有代表性的工作，认为体系结构的层次设计实质上是针对特定优化问题的分解，每一种层次结构都对应着一种优

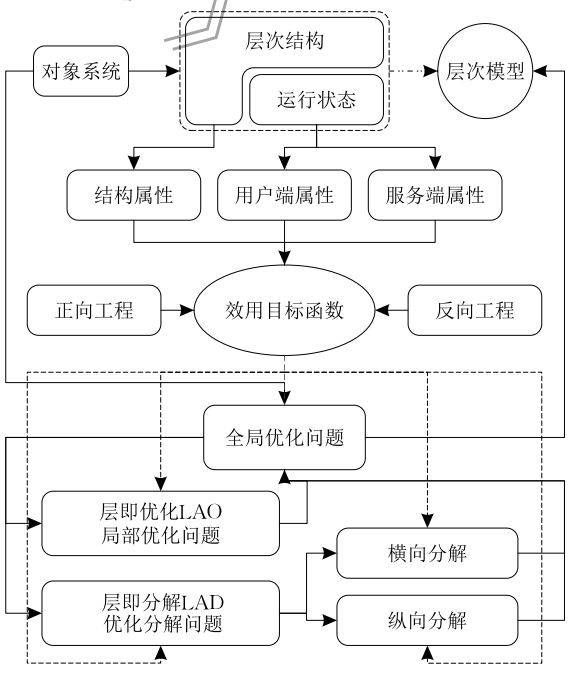

图 2　体系结构层次设计量化模型框架

化分解. Chiang 等人[42]针对网络协议层次结构的广义网络效用最大化问题进行讨论,如式(9)所示.

$$\begin{aligned}&\text{maximize}\quad \sum_s U_s(x_s,P_{e,s})+\sum_j V_j(w_j)\\&\text{subject to}\ \boldsymbol{Rx}\leq \boldsymbol{c}(\boldsymbol{w},\boldsymbol{P}_e),\\&\qquad \boldsymbol{x}\in\mathcal{C}_1(\boldsymbol{P}_e),\ \boldsymbol{x}\in\mathcal{C}_2(\boldsymbol{F})\ \text{or}\ \in\boldsymbol{\Pi}(\boldsymbol{w}),\\&\qquad \boldsymbol{R}\in\mathcal{R},\ \boldsymbol{F}\in\mathcal{F},\ \boldsymbol{w}\in\mathcal{W}\end{aligned}\tag{9}$$

他们将体系结构的层次设计看作一个全局广义效用最大化优化问题的隐含解决方法. 每一层对应一个全局优化问题的子问题,控制包括层间接口变量函数在内的一个决策变量子集,应用本地信息达到部分最优,然后将这些局部算法联合起来应用最优化理论进行全局最优求解.

分层即优化分解框架的含义包含两个方面,层即优化(Layer As Optimizer,LAO)和分层即分解(Layering As Decomposition,LAD). 层即优化将特定层看作一个单独的效用优化问题独立进行求解,关于对应效用函数的构造方法,Zhao 等人[25]和 Cai 等人[26]给出了正向工程和反向工程两种思路. 分层即分解包括横向分解和纵向分解,横向分解是将特定层的优化问题在层内实现为分布式优化问题,部署在不同模块上,纵向分解将功能划分到不同模块. Matni 等人[44]还将分布式优化控制应用到控制系统的层次结构中,将分层即优化分解和分布式优化控制联合考虑,研究层次结构的动态优化控制问题.

3.2 层次构件设计

层次构件设计是一种层次结构设计重要的技术和现象,它将层次结构的某一层或多层子结构的核心功能以单一或较少模块抽象来实现,这些模块将层次结构划分为两部分,这两部分之间没有交互. 构件层需要满足上层不同高级行为的需求,同时可以灵活匹配下层的不同技术. 层次构件设计会使得层次结构在构件层很"窄",形成类似沙漏(Hourglass)形状的结构,这种结构存在于多种层次结构中,也被称为细腰、领结结构. 通过设计合理的交互接口,层次构件设计技术可以使高层面向用户和应用的模块和底层面向资源和通信的模块独立有效发展,从而提高系统的互操作性,减少交互复杂性和故障可能. 层次构件层的形成也可能是系统层次结构在一定条件和环境下演进的现象和结果,如将要提到的 OSI 参考模型.

网格计算[45]体系结构是一个层次结构,体现了层次构件设计. 网格计算的体系结构是一个五层层次结构[33],包括构造层(Fabric)、连接层(Connectivity)、资源层(Resources)、汇集层(Collective)和应用层(Application). 资源层和连接层只由资源和连接协议构成,形成层次构件层,便于不同资源的共享. 这两层协议模块的简化设计可以使他们更容易地被部署在构造层的各种各样类型的资源实现,并且可以更好地被汇集层应用来构成用户和应用需要的服务和功能.

OSI 参考模型为开放式互联系统提供了一个包括物理层、数据链路层、网络层、传输层、会话层、表示层和应用层在内的七层层次结构框架. OSI 七层模型在实际部署中逐渐演化成以网络层为腰部的沙漏结构,如图 3 左所示,使得网络层的设计满足层次构件设计理念. RFC791① 用 45 页文档对网络层 IP 协议的 20 字节结构进行描述和解释,将网络层的核心功能抽象在一个小而精的协议集合中,使得看似简化的网络层蕴含着复杂丰富的逻辑内涵. 这样的设计要求上层各式各样的应用都应该基于 IP 协议技术,同时也允许 IP 地址在下层各式各样的网络上运行[46]. IP 网络是尽力而为和带内管理的,作为网络层在所有网络中传输信息,控制路径和数据路径一致,虽然有许多协议可以和 IP 协议共存,但单一协议的简便性最终使 IP 协议逐渐得到广泛部署.

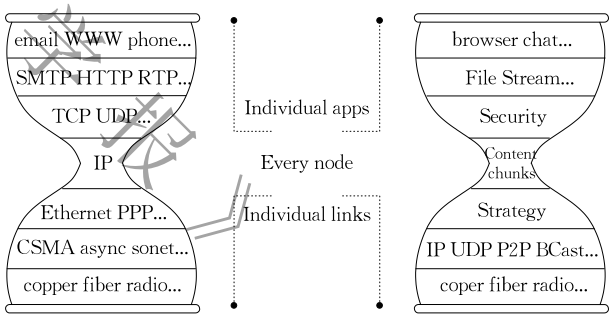

图 3 IP 协议和 NDN 的沙漏模型[47]

Akhshabi 等人[48]提出了 EvoArch 模型评价框架对互联网协议栈的层次结构进行理论分析和评价,尝试对构件层最终形成给予合理解释. 他们将协议栈看成一个层次结构的有向无环图,将协议栈中的协议看作节点分布在协议栈对应的层中,将服务表示为一个有向连接. 这个模型是随时间动态变化的,随时可以有协议节点加入或者离开. 对每个节点 u,定义上层前驱集合 P_u,下层后继集合 S_u,和同层竞争集合 C_u:对于 $p\in P_u$,存在有向连接(u,p);对于 $s\in S_u$,存在有向连接(s,u);对于 $s\in C_u$,$|P_u\cap P_w|/|P_u|\geq c$,其中 c 为竞争阈值. 每一个节点从更高层

① https://tools.ietf.org/html/rfc791

的协议那里获得一个值，并且需要以此为基础和相同层的其他协议节点竞争，演进评价指标和竞争概率定义如下：

$$v(u)=\begin{cases}\sum\limits_{p\in P(u)}v(p), & l(u)<L\\ 1, & l(u)=L\end{cases},$$
$$p_d(r)=\begin{cases}e^{-zr/(1-r)}, & 0<r<1\\ 0, & r\geq 1\end{cases} \tag{10}$$

基于上述模型对网络协议栈随时间变化的演进过程进行分析，给出协议层次模型中沙漏结构细腰“位置”和“宽度”的变化规律、下部明显比上部要小等性质，同时又指出多种协议提供非重复服务宽腰对于体系结构同样重要.

实际上，IP 协议沙漏模型在多媒体、组播、QoS、安全性和移动性上的不足，已经成为重新设计未来网络体系架构的动机. 如 Popa 等人[49]认为可以以 HTTP 协议作为细腰形成新的网络体系架构，即将细腰从三层移到五层. 信息中心网络(ICN)[50]被许多组织提出来解决现有网络体系结构的问题. 其中 NDN(Named Data Networking)[47,51]是来源于 FIA① 的一个具有代表性的项目. NDN 也称为 CCN (Content-Centric Networks)，它的层次结构同样是如图 3 右所示的沙漏结构. 与 IP 网络不同，NDN 根据命名的路由转发方式将数据请求与源/目的网络地址解耦，对网络终端的移动性、数据分组的广播和组播都能够有效支持，同时 NDN 无连接的数据传输方式对数据本身进行数字签名和加密来保证信息的完整性和可靠性[52].

3.3　层次跨层设计

层次结构在特定的系统中需要进一步完善，比如在无线网络体系结构中，无线链路不可靠的特性、无线介质中通信的移动特性、无线信道的广播特性等，都需要通过层次跨层设计来进一步改进.

层次跨层设计(Cross-layer)是指层次结构各层之间建立新的虚拟交互关系. 层次结构在相邻层之间定义严格具有层次关系的接口和服务，通常不允许非相邻层之间的直接交互. 层次跨层设计允许特定层给其他层提供内部接口和参数调用，打破层次结构设计，在已有层次模型的基础上不与相邻层进行交互的设计. 基于提高系统性能的目标，层次跨层设计通常是通过求解一个中心化的跨层联合优化问题，提高系统的特定属性. 层次跨层设计一般是指在一个已有系统层次结构的基础上进行跨层优化，与前文提到的层即分解 LAD 并不完全相同.

Srivastava 等人[53]描述了跨层设计的 6 种结构，如图 4 所示，并且将它们归纳到 4 种层次跨层设计方法：(1) 创建新的接口，图 4A～C；(2) 合并相邻的层，图 4D；(3) 设计没有接口的层耦合，图 4E；(4) 层间纵向校准，图 4F. 根据前文的概念，方法(1)和方法(4)在层次结构逻辑平面创建新的实体或虚拟连接，方法(2)和方法(3)是将层次结构逻辑平面进一步简化.

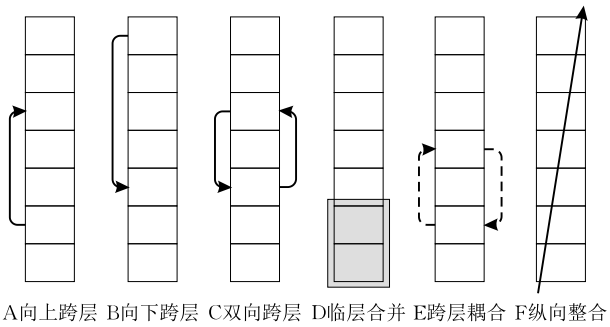

图 4　层次跨层设计[53]

有许多工作都针对自己的特定系统和特定属性进行了不同的跨层设计，其中很大一部分在无线网络体系结构中. Yang 等人[54]将网络层和物理层综合考虑，建立软件定义可编程的控制平面和云计算池，进而在全局视角对移动网络进行控制优化. Barsocchi 等人[55]提出了一种基于垂直跨层集成设计的卫星网络管理架构，将 OSI 模型所有七层的信息共享来进行管理和控制. Fu 等人[56]对自治系统的跨层优化问题给出了一个系统性的框架. 在无线通信的节能优化中，跨层优化设计也得到了广泛应用[57].

网络功能虚拟化(Network Function Virtualization，NFV)是最近出现的一种新技术，它基于现有商用可编程硬件和虚拟化技术，将虚拟网络功能(Virtual Network Function，VNF)的软件实现与底层硬件解耦，提供更加灵活、经济的网络服务[58]. VNF 在不同硬件、网络不同层的实现和部署是 NFV 层次跨层设计的体现. Tofigh 等人[59]认为网络虚拟化跨层信号可以帮助简化大数据的分析过程，并将跨层技术应用到大数据分析系统的 NFV 架构设计中. Carella 等人[60]以 NFV 为基础，在应用层和网络层之间建立联系，通过跨层 API 向客户端和服务端提供网络资源的预约、更改及释放接口，同时提供网络服务和网络流的按需配置，提高网络

① Future Internet Architecture(http://www.nets-fia.net/)

的灵活性.

Rehman 等人[61]还将层次跨层设计应用到软件可靠性评价中,通过功能正确性和时间正确性分析,对运行在非可靠硬件上的软件进行联合可靠性评价与优化.

3.4 层次覆盖设计

层次覆盖设计(Overlay)是由逻辑平面的虚拟节点和逻辑连接组成的虚拟结构,是指层次结构某一层内部结构通过抽象虚拟生成新的简化视图,可以使共存的异构层次体系结构脱离对象系统的固有限制.由于允许异构虚拟结构在一个共享物理层基础上共存,层次覆盖可以提供灵活、差异性的设计,并提高系统安全性和可管理性.

覆盖网(Overlay Network)是一个较为成熟的层次覆盖设计,它覆盖在一个真实存在的基础网络(Underlay Network)之上,以实现现存网络无法实现的网络服务[62].通过软件定义的灵活性[63]及硬件模块虚拟复用[64],虚拟机制可以将物理网络中的同一节点映射到逻辑平面中的多个虚拟节点.抽象机制可以将物理网络中的一条多跳逻辑链路映射到逻辑平面的一条单跳链路(即虚拟连接),根据需要将虚拟节点通过虚拟连接相连形成不同的覆盖网.覆盖网最终形成多个独立的逻辑网络,每一个都可能有不同的地址和转发机制,并且共享一个相同的物理基础架构[65].VXLAN① 是一个典型的覆盖网,可以在三层网络上实现虚拟二层网络.图 5 所示是同一基础网络虚拟产生的两个结构不同的覆盖网,其中一个是星型结构,一个是环状结构,根据需要为上层网络应用服务.

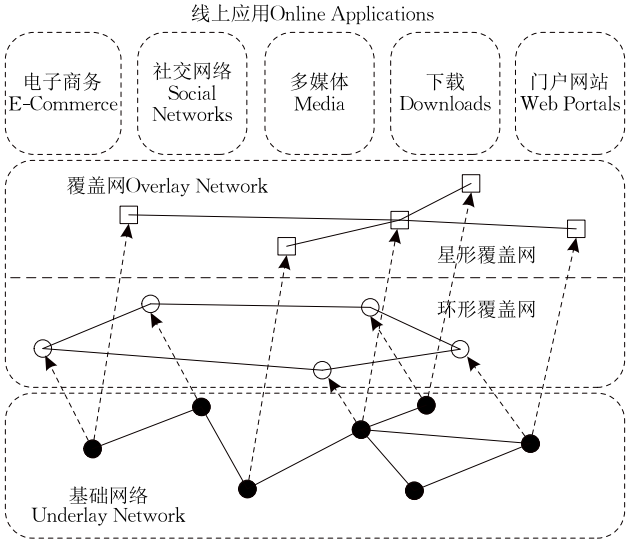

图 5 覆盖网[66]

层次覆盖的功能往往因需求的不同而不同,并且不同功能的层次覆盖也可以共存于同一个层次架构中.以内容缓存、传输路径和内容安全为特征,可以得到三种典型的层次覆盖设计:缓存覆盖网(Caching Overlay)、路由覆盖网(Routing Overlay)以及安全覆盖网(Security Overlay)[66].缓存覆盖网用于传输可缓存(Cacheable)的静态或动态内容,使网络具有更高的性能、可扩展性及可用性,这些可缓存的内容在一定时间尺度上不会发生改变.路由覆盖网用于传输动态或实时流等不易缓存的内容,通过发现更优的虚拟传输路径使得网络具有更低的传输时延、更高的可靠性和吞吐量.安全覆盖网通过向底层基础网络提供安全功能来缓和 DDoS(Distributed Denial of Service)等攻击从而提高安全性.

层次覆盖设计的目标是使多种不同技术更好地融合到层次结构的不同层中,从而为上层多样性的服务提供更好的支持.Lehman 等人[67]从多服务(Multiservice)、多级别(Multilevel)、多技术(Multitechnology)以及多层(Multilayer)的定义出发,提出了一种多层网络体系架构设计框架.该体系架构由应用平面、服务平面、认证与授权平面、管理平面和控制平面五个功能平面覆盖在通用数据平面之上构成,如图 6 所示,使得数据流、服务和虚拟网络环境可以在网络基础架构的所有层动态无缝地转换.

P2P 覆盖是一种典型且应用范围广的层次覆盖设计,已经有很多工作对 P2P 覆盖的模式[63,68]、资源发现[69]、安全[70]以及基于 P2P 覆盖的内容分发技术[71]、网络虚拟环境[72]等方面进行调研和研究.Hu 等人[73]将基于聚类的信誉树引入动态 P2P 系统的层次覆盖架构中,增强了动态网络环境下层次覆盖架构的效率和鲁棒性.Vu 等人[74]将 P2P 覆盖用于大规模分布式系统的分布式负载均衡策略设计,提高了系统的可扩展性.Kleinberg[75]给出分布式结构的网格、层次和集合系统描述模型,来进行分布式查找算法分析.Korzun 等人[76]将这三种模型引入 P2P 覆盖结构,得到了基于聚类(Cluster-based)、基于树(Tree-based)和基于分组(Group-based)的三种系统级层次结构描述模型,并将层次覆盖设计分为纵向(Vertical)和横向(Horizontal)两种设计方法:纵向来看,层次覆盖设计由一个有序或无序的覆盖层集合构成,每一层都是一个完整的覆

① https://tools.ietf.org/html/rfc7348

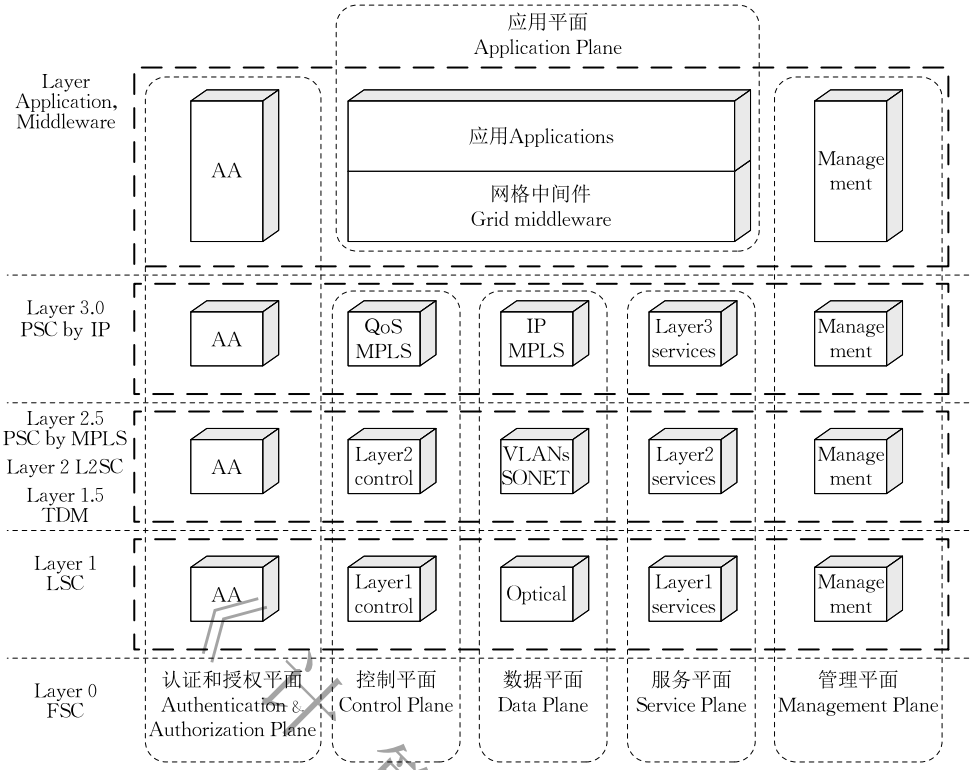

图 6　功能平面层次覆盖分层视图[67]

盖络结构；横向设计是在纵向设计的基础上将每一层分割为不相交的覆盖网结构.

4　层次设计原则和主要机制

基于对计算机系统体系结构层次设计基本概念和研究现状的讨论，我们将系统体系结构层次设计思想归纳为简化和效率两个设计原则，抽象和虚拟两种主要机制，压缩和分解两种实现路径.

4.1　层次设计原则：简化和效率

层次设计原则首先是简化，这源自层次设计最初的设计动机，Dijkstra 认为系统层次结构设计可以对简化系统验证和测试带来巨大的帮助[15]. 与模块化设计原则类似，层次设计将整体系统进行层次划分，使系统各个层次及其之间的相互关系更加简单明了，简化对系统认知、验证、维护、测试、优化的复杂性. 另一方面，在特定计算机系统运行和维护的过程中，出于性能、可靠性等方面属性的优化考虑，需要提高系统的效率，可能使系统更加复杂.

层次构件设计体现的是简化的原则，将层次设计在单层内进行资源和功能简化，降低设计复杂性. 层次跨层设计和层次覆盖设计从两个维度使各层之间的接口和链接设计、状态更加复杂，体现的是效率原则，提高系统效用.

简化和效率在一定条件下可以对应到前面系统的结构(图)复杂性中的客观复杂性 μ 和主观复杂性 $D(S,R)$. 客观复杂性主要考量系统维护管理的复杂性，是一个总量表征，系统越庞大客观复杂性越大. 主观复杂性考量系统与最简参考系统的距离，在以完全图结构作为参考系统时，主观复杂性可以对应通信的便利程度，这时完全图结构是一种主观复杂性最低的情况，任何节点都可以通过单跳通信和网络中的任意其他节点进行通信，从而保证可靠和高效率通信.

参考结构 R 是背景相关的[35]. 对于对等的网络通信结构，如 P2P 网络，网络结构中的通信传输单元的作用和角色基本对称，通信需求对等，主观复杂性可以采用完全图结构作为参考结构 R. 对于如MapReduce 等一些基于 Master-Slave 结构、Fork-Join 结构或神经网络结构的系统层次结构，在程序逻辑结构层面并不需要所有的节点之间进行有效通信，则主观复杂性可以采用 n-分图(n-Partite Graph)结构或部分 n-分图结构作为参考结构. 对于更复杂的系统结构，可能需要用到更加复杂的参考结构 R. 树结构和完全图结构是两种特殊的参考结构，即连通子图可以对原连通图的参考结构进行继承，所以在这两种参考结构下可以进行原图到子图间的映射和复杂性变换.

4.2 层次设计机制:抽象和虚拟

层次设计的是从平面到平面的映射 $\Gamma: P \rightarrow P'$,记为 $P'=\Gamma P$,可以归纳为两种基本设计机制:抽象映射机制 Γ_a 和虚拟映射机制 Γ_v,原平面 P 可以是物理平面或逻辑平面,目的平面 P' 是逻辑平面. 对这两种基本机制叠加和组合方式的不同,可以形成不同的层次结构设计模型和方法,最终对应到不同计算机体系结构的层次设计中. 根据变换前后两个平面实体是否相同及结构变化情况,我们有五种基本映射:单位映射,归一映射,变换映射,抽象映射,虚拟映射. 其中,单位映射 Γ_I 是指将一个平面的结构映射为相同的结构,即 $G(P')=G(P)$,且目标平面 P' 所包含的每个节点的功能、属性均与原平面对应节点相同;变换映射 Γ_t 是指将一个平面的结构映射为相同的结构,即 $G(P')=G(P)$,但是至少存在一个节点的属性和功能与原平面对应节点不同;归一映射 Γ_0 是指将一个平面结构映射为单点结构,即 $G(P')=\{v\}$;抽象映射 Γ_a 和虚拟映射 Γ_v 的定义在下文详细给出.

4.2.1 抽象机制

抽象的概念与模型的建立密不可分,是计算机系统体系结构建模和研究的重要途径[77]. 一个系统的抽象蕴含两重含义,一方面它将有关的特定属性进行抽象,一方面它也忽略了一些属性. 这引出了抽象的两个方面:信息忽略和信息隐藏. 信息忽略在对主要矛盾构建形式化数学模型的同时排除无关紧要的细节. 在数学科学中,通常以尽可能少的抽象将与研究对象的无关属性剔除,它既可以用于发现研究对象的普适性质,也可以用来证明研究对象的特定性质与研究问题并不相关. 计算机系统体系结构设计的抽象属于信息隐藏的范畴,它隐藏但是并不忽略相邻层会话中对于层次设计及上下文无关紧要的细节. 因为隐藏的细节在该层功能的实现中是不必要的,但是在其他层的功能实现中有可能是必需的,这个观察蕴含在计算机系统体系结构设计中抽象和解释机制的实现中. 如计算机系统的运行过程,将动态电子事件组合,用程序设计语言文本化表示,其中常用的各种形式化程序设计语言都是类似的抽象工具,应用编译器、汇编器、连接器将基本逻辑进行隐藏.

基于上面的论述,我们对抽象机制有如下定义.

定义 9(抽象机制,Abstraction). 如图 7 所示,抽象机制是一个映射 Γ_a,它将原平面结构 $G(P)$ 的一个子结构 $G_a(P)$ 映射为目标平面结构 $G(P')$ 的一个节点实体 v_a. 抽象机制并不改变结构的连通特性,因此,如果 $G_a(P)$ 包括了 $G(P)$ 中多个连通分支,则等价于抽象为多个节点实体,分别存在于相应连通分支中.

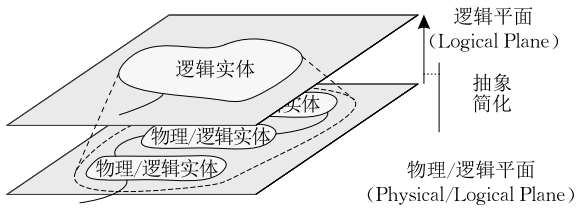

图 7 抽象映射

抽象机制用于在目的平面建立简化的结构,隐藏结构模块背后的实现细节和信息,减少模块、层之间的相互作用,简化结构背后部件的调用. Γ_a 直观上是一个收缩映射,目标平面的平面复杂性及结构复杂性会降低. 抽象机制的目标可以归纳为:(1)使目标平面结构更加简洁并且在时间尺度上长期稳定;(2)使目标平面结构内模块、层对原平面不同逻辑功能提供灵活便利的封装、实现和调用. 在本文的定义中,模块、层和层次分别和各个级别的抽象一致. 合理的抽象可以通过对接口的封装和定义更好地隐藏相关模块背后的实现细节,也可以优化减少模块、层之间的交互需求,但无法保证提高系统层次结构的性能或效率.

4.2.2 虚拟机制

与抽象机制类似,虚拟机制的概念定义如下.

定义 10(虚拟机制,Virtualization). 如图 8 所示,虚拟机制是一个映射 Γ_v,它将原平面结构 $G(P)$ 的节点实体 v_v 映射为目标平面结构 $G(P')$ 的子结构 $G_v(P)$,$G_v(P)$ 中每个节点至少继承 v_v 的一个连接.

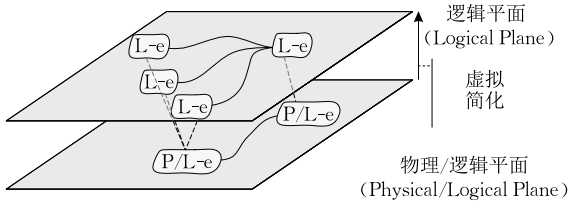

图 8 虚拟映射

虚拟机制用于在目的平面建立更加细致灵活的结构,实现更加复杂的功能,平面 P 的一个实体可能映射到平面 P' 的一个或多个相同或不同的实体. Γ_v 直观上是一个扩张映射或变换映射,目标平面的平面复杂性及结构复杂性不会降低.

常见的虚拟机制是从物理平面到相邻逻辑平面的映射,是对系统和部件的物理观察转化为逻辑观

察的映射过程，如虚拟机，虚拟网络等. 它包括变换、复制和隔离 3 个目标. 变换是将资源进行接口变换在逻辑平面实现不同功能；复制是生成资源的多个逻辑复制品在逻辑平面同时运行；隔离是指逻辑平面映射到的每一个逻辑部件都可以独立的运行和管理，相互隔离. 所以，与抽象机制强调功能的封装不同，虚拟机制更强调平面映射的转变进而实现更加灵活的功能.

虚拟机制在计算机系统中已经得到广泛应用，虚拟化技术已经成为管理计算架构的主流技术[78]. 网络虚拟化[79]是虚拟化技术中的重要组成部分，它将传统的网络服务提供商(ISPs)解耦成网络基础设施提供商(INPs)来提供物理基础设施以及服务供应商(SPs)来创建虚拟网络(VNs)[80]. 服务提供商可以通过虚拟机制在异构网络架构中部署可定制的端到端服务[81]. 虚拟机制还可以在网络资源管理架构中提供容错管理和移动支持，进而提高资源管理的有效性[82].

4.3　层次设计路径：压缩和分解

在层次设计原则和机制的基础上，可以对目标计算机系统进行层次设计. 层次设计实现路径是指层次设计的方法和设计过程，本小节介绍压缩和分解两个层次设计实现路径，通常需要将它们结合起来. 这两种实现路径蕴含着计算机系统体系结构设计和评价的两种思路：在进行系统设计时，通常采用自顶向下逐步分解求精的思路；在对系统进行评价时，通常采用自底向上逐步压缩抽象的思路. 两种层次设计路径的形式化定义如下.

定义 11(压缩过程，Aggregation).　自底向上逐步压缩，压缩过程是一个由初始平面 P 到可接受目标平面 P' 的映射序列，$P'=\Gamma_n\Gamma_{n-1}\cdots\Gamma_1 P=\Gamma^0 P$.

压缩过程的思路是自底向上的，通过选择三个合适的 Γ^0，可以得到图 1 中的结构. 需要注意，如果迭代应用的过程中，$P=P_{MR}$，$P'=P_{WH}$，那么 Γ^0 并不总能分解为合适的三部分，使得不同分段映射对应到图 1 中的结构. 与压缩过程相对，系统体系结构的层次设计同样可以先从逻辑平面最顶端的一个抽象的整体系统开始，先进行粗略的层次设计，并逐步细化，直到得到整个系统的层次设计.

定义 12(分解过程，Decomposition).　自顶向下逐步分解，分解过程是对初始平面 $P=\{v_0\}$ 到可接受目标平面 P' 映射序列，$P'=\Gamma_n\Gamma_{n-1}\cdots\Gamma_1 P=\Gamma^0 v_0$.

分解过程的思路是自顶向下的，它与压缩过程形式类似，但压缩过程中的映射多为压缩映射，而分解过程中的映射多为扩张映射. 复杂的结构可以由多种层次结构来表示和构成，我们需要使用压缩过程和分解过程来帮助我们选择合适的抽象、虚拟映射，最终得到可以接受的模块、层以及层次的结构.

压缩过程是从一个可以工作的简单架构开始，这个架构至少可以满足一些关键需求，这可以看作物理平面，随后应用抽象和虚拟机制映射到逻辑平面进行复杂性等方面的改进，并从可能会出现的问题中总结经验，甚至可能影响到最初物理平面资源的部署和结构. 分解过程则对应着另一种思路，首先对初始平面的整体系统进行粗略的功能性划分，并以之为基础尝试逐步细化到可接受结构. 这两种实现路径均体现了分治法和层次化解决问题的思路.

5　层次设计结构评价

系统体系结构层次设计属性评价的目的主要有三个：选择、改进和设计[83]. 即在许多系统体系结构层次设计中选择一个最适合需要的系统体系结构层次设计，针对特定属性对已有系统体系结构层次设计进行改进，提出可能的更好系统体系结构进行设计方案. 以第 4 部分归纳出的层次设计原则和机制为基础，我们对计算机系统体系结构层次设计的复杂性和性能进行评价，一方面可以凭借这样反向工程的思路对已有层次设计机制、现象、系统提供量化解释和理论支撑，另一方面还可以对层次设计新技术进行理论优化.

5.1　层次结构复杂性评价

基于第 2 节给出的层次设计基本概念与定义，我们对层次设计进行复杂性评价. 直观上针对层次设计简化的原则，层次设计相对原系统整体设计的复杂性应该减小. 抽象机制是系统从最底层平面形成层次结构的主要机制. 这里从抽象映射 Γ_a 对目标平面 P' 和原平面 P 复杂性的影响出发，讨论系统体系结构设计的简化原则.

对前文复杂性的定义，取 $\mu(G)=|V|\times|E|$. 我们给出两个假设，假设 1 使得原平面和目的平面的结构在映射前后参考结构属性不变；假设 2 使得初始平面并不是一个中间平面，这一点我们会在后面进行讨论.

假设 1.　$R=(V_r,E_r)$ 为连通结构 $G=(V,E)$ 的参考结构，$n_r=n=|V|$，$m_r=f_e(n)$，假设参考结构取：(1) 树结构，即 $f_e(n)=n-1$；或(2) 完全图结

构，即 $f_e(n)=n(n-1)/2$. 这两种参考结构的属性不会因对应结构逻辑功能、机制映射的不同而改变，且 $\mu(MCS(S,R))=\min\{\mu(S),\mu(R)\}$. 在一个系统中，只取其中一种.

假设 2. 原平面 P 为最底层平面，如图 1 中的物理平面.

针对单位映射 Γ_I、归一映射 Γ_0 和抽象映射 Γ_a 对映射前后系统复杂性的影响，下面给出了 3 个定理和相关的中间引理.

引理 1. 单节点结构复杂性 $K(v)=K(v^{-1})$，即孤立系统的复杂性等于其内部结构的复杂性.

证明见附录.

定理 1(单位映射不变定理). 若映射为单位映射 Γ_I，$G(P)=G(P')$，则 $K(P')=K(P)$.

证明. 结合引理 1.

$$\begin{aligned}K(P')&=\sum_{cc\in CC(P')}\Big[K'(cc)+\sum_{v\in cc}K(v^{-1})\Big]\\&=\sum_{cc\in CC(P')}K'(cc)+\sum_{cc\in CC(P')}\sum_{v\in cc}K(v^{-1})\\&=\sum_{cc\in CC(P)}K'(cc)+\sum_{cc\in CC(P)}\sum_{v\in cc}K(v)\\&=\sum_{cc\in CC(P)}K'(cc)+\sum_{cc\in CC(P)}\sum_{v\in cc}K(v^{-1})\\&=\sum_{cc\in CC(P)}\Big[K'(cc)+\sum_{v\in cc}K(v^{-1})\Big]\\&=K(P).\end{aligned}$$ 证毕.

定理 2(归一映射不变定理). 若映射为归一映射 Γ_0，$G(P')=(\{v_0\},\Phi)$，则 $K(P')=K(P)$.

证明. 结合引理 1.

$K(P')=K(v_0)=K(v_0^{-1})=K(P)$. 证毕.

引理 2. 结构 $G_0=(V_0,E_0)$ 有 $l=|CC(G)|$ 个连通分支 $G_j=(V_j,E_j)$，$n_j=|V_j|$，$m_j=|E_j|$，$j\in[0,l]$，$n_0=\sum n_j$，$m_0=\sum m_j$，参考结构满足假设 1、假设 2 条件，对于式(3)给出的结构复杂性定义，有如下关系：

$$K'(G_0)\geq\sum_l K'(G_j)\tag{11}$$

即 G_0 为连通图时复杂性最大，证明见附录.

引理 3. 对于结构 $G_j=(V_j,E_j)$，$j=0,1,2$，其中 G_0，G_2 为连通结构，$n_j=|V_j|$，$m_j=|E_j|$，参考结构满足假设 1、假设 2 条件，当 $m_0\geq m_1+m_2$，$n_0\geq n_1+n_2-1$ 时，对于式(3)给出的结构复杂性定义，有如下关系：

$$K'(G_0)\geq K'(G_1)+K'(G_2)\tag{12}$$

证明见附录.

定理 3(抽象简化定理). 在假设 1、假设 2 条件成立时，若映射为抽象映射 Γ_a，则 $K(P')\leq K(P)$.

证明. 因为抽象映射不影响结构的连通性，所以抽象映射 Γ_a 作用于 $G(P)$ 时，是对部分连通分支集合 $CC_a(P)\subseteq CC(P)$ 分别进行抽象映射，到平面 P' 对应连通分支，属于连通分支集合 $CC_a(P')\subseteq CC(P')$，对其余的连通分支集合 $CC_I(P)\subseteq CC(P)$ 分别进行单位映射，到平面 P' 对应连通分支，属于连通分支集合 $CC_I(P')\subseteq CC(P')$，并且 $CC_a(P)\cup CC_I(P)=CC(P)$，$CC_a(P')\cup CC_I(P')=CC(P')$.

情况(1)对 $CC_I(P)$，根据定理 1，可以得到

$$\sum_{cc'\in CC_I(P')}K(cc')=\sum_{cc\in CC_I(P)}K(cc).$$

情况(2)对 $cc\in CC_a(P)$ 到 $cc'\in CC_a(P')$ 的抽象映射，记 $G_0=cc$，$G_2=cc'$，对于抽象部分 $G_1\subseteq G_0$，$v_0\in G_2$，记 $v_0^{-1}=G_1$，$l=|CC(G_1)|$，易知 G_0，G_1，G_2 满足引理 3 条件，结合引理 1～3，可以得到

$$\begin{aligned}K(cc')&=K(G_2)=K'(G_2)+\sum_{v\in G_2}K(v^{-1})\\&=K'(G_2)+\sum_{v\in G_2/v_0}K(v^{-1})+K(v_0^{-1})\\&=K'(G_2)+\sum_{v\in G_0/G_1}K(v)+K(G_1)\\&=K'(G_2)+\sum_{v\in G_0/G_1}K(v^{-1})+\\&\quad\sum_l K'(G_j)+\sum_{v\in G_1}K(v^{-1})\\&\leq K'(G_2)+\sum_{v\in G_0/G_1}K(v^{-1})+\\&\quad K'(G_1)+\sum_{v\in G_1}K(v^{-1})\\&=K'(G_2)+K'(G_1)+\sum_{v\in G_0}K(v^{-1})\\&\leq K'(G_0)+\sum_{v\in G_0}K(v^{-1})\\&=K(G_0)=K(cc).\end{aligned}$$

其中第 3 行到第 4 行等式应用引理 1，第 4 行到第 5 行不等式应用引理 2，第 6 行到第 7 行不等式应用引理 3. 综上情况(1)、情况(2)所述，

$$\begin{aligned}K(P')&=\sum_{cc'\in CC_I(P')}K(cc')+\sum_{cc'\in CC_a(P')}K(cc')\\&\leq\sum_{cc\in CC_I(P)}K(cc)+\sum_{cc\in CC_a(P)}K(cc)\\&=K(P).\end{aligned}$$ 证毕.

关于三个定理，是符合直观的：单位映射 Γ_I 将原平面的结构“复制”到目标平面，结构和属性没有改变，复杂性评价指标应不发生改变；归一映射 Γ_0 将原平面的结构映射成为只有一个实体节点 v_0 的目标平面结构，新平面的复杂性应全部蕴含在这个

节点，复杂性评价指标应等同于该节点内部结构即原平面结构的复杂性评价指标；抽象简化定理尝试解释进行抽象、模块化、封装的动机——“简化”的量化内涵，即使系统的复杂性降低，也是符合直观的.

关于定理 3 抽象简化定理和假设 2，我们做如下讨论. 如图 9 所示，记最底层的物理平面为 P_0，最高层的单一结点对象系统平面为 P_4，通过不同抽象虚拟机制映射得到三种可能的系统层次结构设计方案分别为 P_1、P_2、P_3. 由定理 2 知，$K(P_0)=K(P_4)$，由定理 3 知，$K(P_4)=K(P_0)\geq K(P_1)$，$K(P_2)$，$K(P_3)$. 即在给定物理平面后，对于平面 P_0，任何自底向上的压缩过程都会使系统复杂性评价指标减小；对于平面 P_4，任何自顶向下的分解过程都会使系统复杂性评价指标减小.

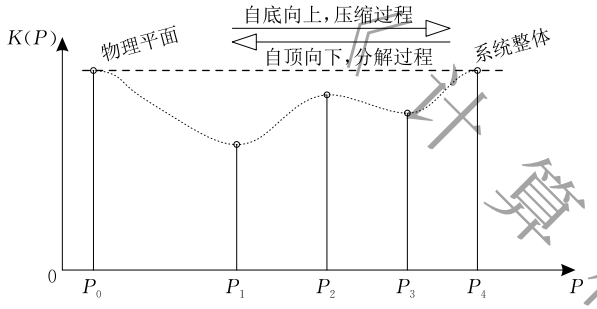

图 9　不同层次结构设计复杂性评价指标示意图

如果没有假设 2，即对于任一层次结构设计结果，压缩过程和分解过程对于系统复杂性评价指标的影响是未知的，即可能存在图 9 中经过压缩过程后 $K(P_2)\geq K(P_1)$ 的情况，经过分解过程后 $K(P_3)\leq K(P_2)$ 的情况. 这是因为进行压缩过程时的原平面和分解过程的目的平面涉及到的结构可能有着复杂的交互连接关系，并不相互独立，合并后、分解前的内部复杂性评价指标数值更大. 对于一个有限物理平面结构的划分方法是有限的，所以一定存在一个层次设计方案 $P_{\min}$ 使得系统层次结构的复杂性评价指标最小，这种设计方案平面，在我们定义的复杂性评价指标意义下，$P_{\min}$ 是对象系统最优简化的体系结构层次设计方案. 所以，可以将体系结构层次设计简化问题描述如下：

$$\underset{\Gamma^0}{\text{minimize}}\ K(\Gamma^0 P_0) \tag{13}$$

其中，P_0 为物理平面时，Γ^0 为压缩过程；P_0 为系统整体时，Γ^0 为分解过程.

5.2　层次结构性能评价

系统的性能取决于多种因素，包括系统的配置和系统负载两个方面[83]. 系统的配置指系统体系结构及构成系统的成分、数量、能力等，系统的负载指工作负载和工作方式. 这里我们考量系统体系结构层次模型对性能的影响.

计算机系统体系结构层次模型的服务过程可以看作特殊的客户端-服务器(Client-Server，C-S)模型进行效用评价，这里我们考虑效用的两个基本评价指标形式：响应时间和吞吐量. 可以将每一个层次结构看成三个具有不同性质的层，包括客户端层、客户端/服务器层、服务器层，如图 10 所示. 每一层可能有多个模块，并且同一层内的不同模块可能放置于不同的地理位置. 客户端层的节点只发送请求；客户端/服务器层的节点处理来自上层或本层的请求，并向本层或下层发送请求；服务器层节点只处理来自客户端/服务器层的请求. 图 10 中实线有向边表示请求连接，虚线有向边表示应答方向示意，因为请求/服务层可以互相发送请求，所以可能存在循环调用的情况.

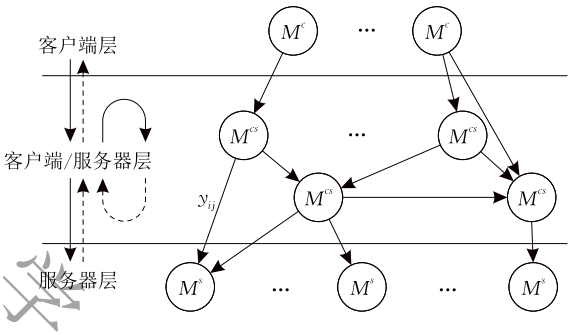

图 10　层次结构性能评价 C-S 模型

记 t_i 为节点 i 的响应时间，s_i 为节点 i 的执行时间，w_{ij} 为节点 i 对节点 j 请求到达时的等待时间，y_{ij} 为节点 i 执行过程中对节点 j 发送请求的平均数量，f_i 为节点 i 稳定状态下的吞吐量，则有

$$t_i = s_i + \sum_j y_{ij} w_{ij},$$

$$f_j = \sum_i y_{ij} f_i \tag{14}$$

在之前的工作[84]中，我们应用基于随机集结网络的 C-S 模型对目标计算模式的层次结构进行建模，并对响应时间 $t_i>0$ 和稳定状态吞吐量 $f_i<\infty$ 进行迭代求解. 假定我们取效用函数 $U_i^t(v_i)=1/t_i$，$U_i^f(v_i)=f_i$，则一定存在一个层次设计方案 $P_{\min}$ 使得系统体系结构层次设计的效用 $U(P)=\sum_i U_i$ 最大，则在相应的响应时间和吞吐量效用定义下，$P_{\min}$ 是系统最优效用的体系结构层次设计方案. 所以，和已有效用最大化工作类似，可以将体系结构层次设计效用问题描述如下：

$$\underset{\Gamma^0}{\text{maxmize}}\, U(\Gamma^0 P_0) = \sum_i U(v_i) \qquad (15)$$

其中，$v \in V(G(\Gamma^0 P_0))$，$U(\cdot)$为目标效用函数.

5.3 讨 论

以上关于系统体系结构层次设计机制复杂性、性能的评价与讨论，可以看出本文层次结构设计模型框架是有效的，在一定程度上对计算机系统体系结构层次设计问题的本质探索进行了尝试，将层次设计的原则归纳为简化和效率，并给出相应复杂性和性能的评价框架，得到的结果是有益的. 当然，这样的评价框架和本质探究仍然是较为抽象和初步的，还可以进一步细化、针对不同对象系统进行具体化，从而更好地应用到实际系统中，应用得到的结论，对对象系统体系结构层次设计进行指导.

6 层次设计经典系统例子

在计算机系统发展演进的过程中，形成了体系结构、网络和计算等多种层次结构设计模式，它们蕴含着层次设计的原则、机制和设计方法. 我们以层次设计的角度对三个经典计算机系统例子——超级计算机、软件定义网络和云计算进行介绍.

6.1 超级计算机

超级计算机(Supercomputer)系统是计算机系统体系结构发展的重要方向，主要面向高性能计算提供强大的计算能力和海量的存储空间支持. 目前超级计算机的主要实现途径是大规模并行处理(Massively Parallel Processing，MPP)，充分利用时间和空间并行，将同一任务的不同部分分别交给多个处理器(Processor)、协处理器(Coprocessor)处理的过程. 处理过程中各处理器应用各自操作系统和内存，通过数据通路互连传递信息. 天河超级计算机是一个著名的超级计算机系统，它的层次结构如图 11 所示，是包含虚拟机、物理节点、层叠网络、物理网络的多层层次结构.

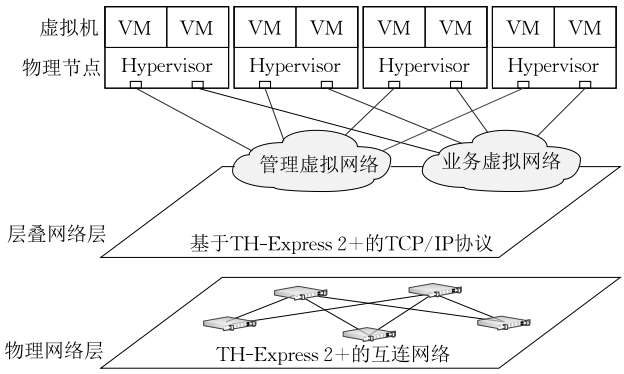

图 11 超级计算机层次结构[85]

廖湘科等人[85]就面向大数据计算的超级计算机体系结构设计进行了讨论，提出了一些改进设计方案. 例如：(1)通过硬件上将 CPU 与 GPU 放置在同一个芯片等策略降低 CPU 到协处理器的数据延迟；(2)通过建立处理器与二级缓存之间的数据通道优化数据缓存提高吞吐量. 在本文定义的概念中：(1)应用了物理平面的抽象机制，将 CPU 和协处理器抽象成一个模块进行物理平面设计，并且在模块内降低数据延迟；(2)应用了层次跨层设计，打破缓存层次结构，通过建立新的接口和连接提高系统的性能.

6.2 软件定义网络

SDN 是当前网络体系结构研究的热点[86-87]，它的层次结构如图 12 所示：从网络功能的视角来看，包括管理平面、控制平面和数据平面 3 层；与之对应，从系统体系结构设计的视角来看，由网络应用、控制器和网络基础设施 3 层构成. 管理平面和控制平面之间通过北向接口(NorthBound Interface，NBI)进行交互，控制平面和数据平面之间通过南向接口(SourthBound Interface，SBI)进行交互. SBI 将控制平面与数据平面的功能分离，使控制平面应用 Openflow[88]、OVSDB①、ForCES② 等多种控制数据平面接口(Control Data Plane Interface，CDPI)对数据平面的物理设备和虚拟设备(如 Open vSwitch，vRouter)进行管理. NBI由一系列接口软件构成.

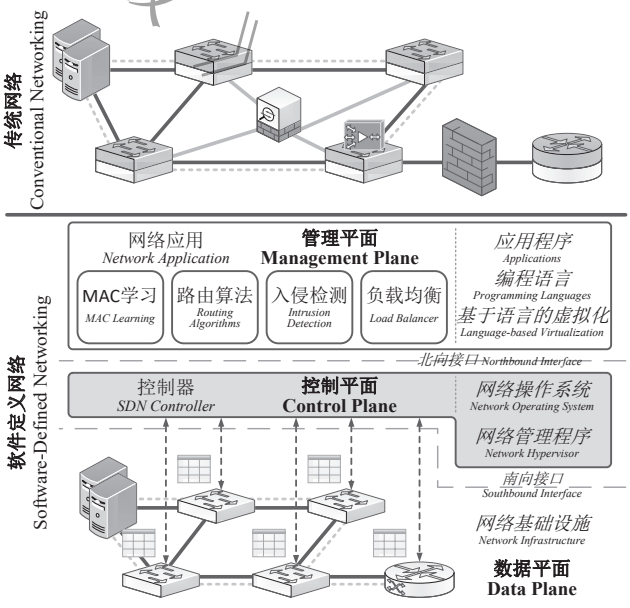

图 12 软件定义网络层次结构[89]

① https://tools.ietf.org/html/rfc7047
② https://tools.ietf.org/html/rfc5810

Kreutz 等人[89]更加细致地将 SDN 层次结构分为网络基础设施、南向接口、网络监控器、网络操作系统、北向接口、基于语言的虚拟化、编程语言、网络应用共 8 层. 其中，南向接口、网络操作系统、北向接口以及网络应用层普遍存在于 SDN 的部署实践中，网络管理程序、基于语言的虚拟化层则只是存在于特定的 SDN 部署实践中.

与传统网络控制模式不同，SDN 将控制逻辑从数据层分离，模块化封装成各种控制策略服务. 这样的设计允许管理平面根据不同需求通过 NBI 灵活编程部署相应网络控制应用，并通过 CDPI 安装到数据平面不同的网络设备中. 不同平面之间的行为通过标准接口约束，使得各平面内部可以异构部署和独立演进[90].

控制平面具有全局网络视图，是网络控制逻辑的抽象. 与 IP 协议类似，控制平面能够灵活支持上层多样的应用，同时也要求控制平面服务能够在多样的数据平面网络基础设施上运行，是层次构件设计的体现. 数据平面通过控制平面逐流形成控制逻辑，等价于在数据层对不同流形成对应的覆盖网，对每条流进行控制优化，可以提高网络的可管理性. 控制平面一般由集中式的控制器构成，出于性能的考虑，形成了多线程[91]、分布式[92]和层次式[93]等控制机制，Hu 等人[94]还对控制平面的可扩展性进行了模型研究.

6.3　云计算

云计算是一种网络计算模式，能够通过网络以便利、按需、可计量的方式弹性地获取网络、服务器、存储、应用和服务等计算资源. 这些资源来自一个可配置的共享资源池，并能够对资源动态灵活地获取和释放[95]. 如图 13 所示，云计算层次结构模型包括云计算客户端、软件即服务(SaaS)、平台即服务(PaaS)以及基础设施及服务(IaaS). Client 层提供个性化的应用计算环境；SaaS 层集中设置、管理和运行应用软件，并接受 Client 层的远程访问和调用；PaaS 层允许用户对应用进行开发、部署与管理；IaaS 向用户提供处理、存储、网络等基本计算资源来运行任意应用程序. 云计算的设计目标是更好地使用分布的资源，提高吞吐量及求解大规模计算问题. 云计算有多种部署模式，包括私有云、公有云、社区云和混合云.

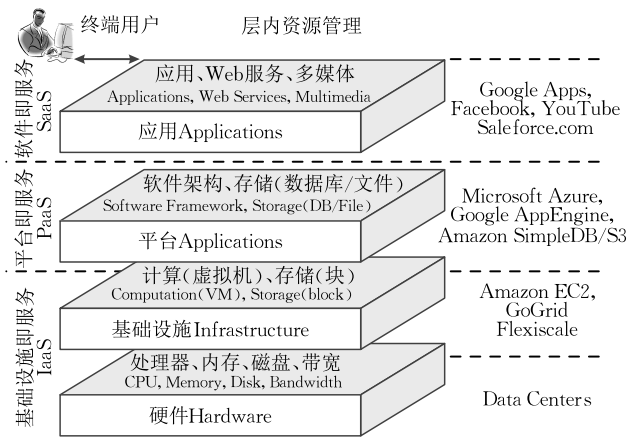

图 13　云计算层次结构[96]

基于组件的软件工程(CBSE)[97]及面向服务架构将一些松耦合、黑盒式的组件进行组合，发布为明确定义的服务来实现业务流程. 从服务的角度来看，云计算不同级别的资源池以及 SOA 基于组件生成的各种服务，核心就是层次设计中的模块化抽象和虚拟复用，与层次设计的抽象机制和虚拟机制对应，使系统的逻辑管理更加简化，在提高复用性的同时可以更灵活地应对需求的变化，确保客户需求、软件开发的效率和可靠性.

6.4　复杂性和性能评价思路

从 C-S 模型的角度来看，计算机系统可以粗略划分为用户端和服务端两层，用户端将任务需求提交给服务端，由服务端进行服务. 以上文对三个例子的介绍为基础，图 14 给出了超级计算机、软件定义网络和云计算的宏观 C-S 模型.

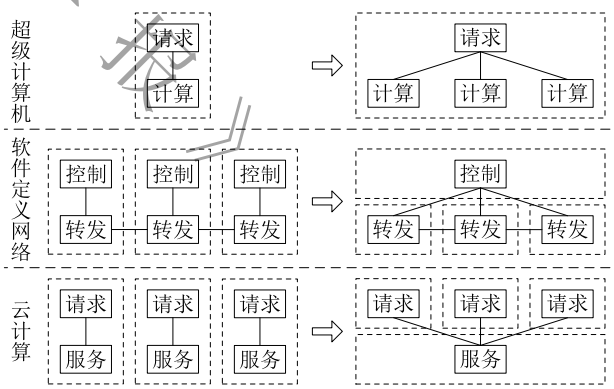

图 14　三种层次结构设计宏观视图

可以看出，三种体系结构层次设计分别代表了层次设计机制的三种典型应用. 超级计算机充分利用并行以及大规模计算资源，将请求分布到计算层的多个计算模块中；软件定义网络将分布式的控制策略平面，抽象成为统一全局控制器，独立形成控制策略平面；云计算将传统计算机的本地服务部分或全部放在云端，形成云端的“一切皆服务”[98].

本文的体系结构描述是较高抽象级别的，对设计和实现过程中可能出现的某些限制考虑的比较

少.在宏观 C-S 模型层次结构的基础上,可以针对对象系统具体细节进行细化,进而应用我们给出的层次结构复杂性和性能评价框架,理论上给出简化和效用优化.

7 总结与未来工作

计算机系统体系结构层次设计是计算机科学技术、网络和应用发展的重要组成部分.虽然在体系结构发展过程中形成了一些设计和量化原理,而且在现有系统体系结构设计中都隐含着一些层次设计的方法,但现有的层次设计缺少对层次设计机制内涵的理解分析,以及统一的层次设计模型框架和评价指标.本文从上述问题入手,给出了一个计算机系统体系结构层次设计的模型框架.我们在这个框架下:(1)给出了层次设计基本概念、基本原则和主要机制的语义及定义;(2)从四个方面对计算机系统层次设计研究现状进行综述,并在此基础上对计算机系统体系结构层次设计设计原则、设计机制和设计路径进行讨论;(3)给出了层次结构复杂性和性能的评价指标和评价方法,就不同层次设计机制对系统复杂性的影响进行讨论,得到了一些符合直观的有益结论;(4)对三个计算机体系结构层次设计经典系统例子进行介绍,并对他们所应用到的层次设计原则和机制进行对应讨论.我们模型框架与系统实现细节相比更加抽象,集中精力在计算机系统体系结构设计的主要特征,因此评价框架结论可以应用到更广范围的系统中.

同时,我们的研究仍有很多后续和未来工作可以进行探索:

模型框架完善方面:(1)评价指标体系研究:根据实际系统需求,给出除复杂性和性能之外的其余重要评价指标的语义和形式化定义,形成层次结构模型评价指标体系;(2)参考结构的适用性研究:对复杂性评价指标中参考结构的性质进行深入研究,探索更一般参考结构下各类映射对层次结构复杂性评价的影响;(3)映射的有序表达研究:目前映射形成序列的过程仍需要人为操作进行,如何对一个结构进行的不同映射方法的序进行有效表达,对更好地进行层次设计和属性优化十分重要;(4)模型适用性研究:对模型框架和指标体系在实际系统设计中具体的指导和优化影响进行研究;

模型评价求解方面:(5)C-S 模型在非 DAG 模型下的求解方法研究;(6)多目标评价研究:本文仅对复杂性和性能分别进行单目标的评价,而且文中复杂性建模原本就包括了客观复杂性和主观复杂性两个子指标,整体模型框架都蕴含着多目标的评价和优化问题,可以进一步进行多目标优化研究.

模型框架应用方面:(7)层次设计机制的反向工程应用:以该模型框架为基础,对计算机体系结构层次设计机制、方法进行抽象建模和评价,一方面可以凭借这样反向工程的思路对已有层次设计机制、技术、系统提供量化解释和理论支撑,另一方面还可以对层次设计新机制和技术进行理论优化;(8)知识总结表示和应用:整合模型框架知识,以便更方便、清晰地对体系结构模型设计、描述进行指导.它们也是我们下一步工作的方向.

参 考 文 献

[1] Hennessy J L, Patterson D A. Computer Architecture: A Quantitative Approach. 4th Edition. Amsterdam, Netherlands: Elsevier, 2011

[2] Harrison N B, Avgeriou P. How do architecture patterns and tactics interact? A model and annotation. Journal of Systems and Software, 2010, 83(10): 1735-1758

[3] Bass L. Software Architecture in Practice. 2nd Edition. Delhi, India: Pearson Education India, 2003

[4] Kivelä M, Arenas A, Barthelemy M, et al. Multilayer networks. Journal of Complex Networks, 2014, 2(3): 203-271

[5] Broy M. Can practitioners neglect theory and theoreticians neglect practice?. IEEE Computer, 2011, 44(10): 19-24

[6] Johnson P, Ekstedt M, Jacobson I. Where's the theory for software engineering? IEEE Software, 2012, 29(5): 94-95

[7] Denning P J. The locality principle. Communications of the ACM, 2005, 48(7): 19-24

[8] Amdahl G M. Validity of the single processor approach to achieving large scale computing capabilities//Proceedings of the Spring Joint Computer Conference. Atlantic City, USA, 1967: 483-485

[9] Eden A H, Kazman R. Architecture, design, implementation //Proceedings of the 25th International Conference on Software Engineering. Portland, USA, 2003: 149-159

[10] Clements P, Garlan D, Bass L, et al. Documenting Software Architectures: Views and Beyond. New York, USA: Pearson Education, 2002

[11] Tyree J, Akerman A. Architecture decisions: Demystifying architecture. IEEE Software, 2005, 22(2): 19-27

[12] Tang A, Han J. Architecture rationalization: A methodology for architecture verifiability, traceability and completeness// Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05). Maryland, USA, 2005: 135-144

[13] Tang A, Babar M A, Gorton I, et al. A survey of architecture design rationale. Journal of Systems and Software, 2006, 79(12): 1792-1804

[14] Doyle J C, Csete M. Architecture, constraints, and behavior. Proceedings of the National Academy of Sciences, 2011, 108 (Supplement 3): 15624-15630

[15] Dijkstra E W. The structure of the "THE"-multiprogramming system. Communications of the ACM, 1968, 11(5): 341

[16] Zimmermann H. OSI reference model-the ISO model of architecture for open systems interconnection. IEEE Transactions on Communications, 1980, 28(4): 425-432

[17] Erl T. Service-Oriented Architecture: Concepts, Technology, and Design. Delhi, India: Pearson Education, 2005

[18] 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

[19] McKeown N. Software-defined networking. INFOCOM Keynote Talk, 2009, 17(2): 30-32

[20] Tan L, Wang N. Future internet: The Internet of Things// Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE). Chengdu, China, 2010, V5-376: V5-380

[21] Faro A, Giordano D, Spampinato C. Integrating location tracking, traffic monitoring and semantics in a layered ITS architecture. IET Intelligent Transport Systems, 2011, 5(3): 197-206

[22] Agarwal A, Shankar R. A layered architecture for NOC design methodology//Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing Systems. Phoenix, USA, 2005: 659-666

[23] Zhu Y, Bavier A, Feamster N, et al. UFO: A resilient layered routing architecture. ACM SIGCOMM Computer Communication Review, 2008, 38(5): 59-62

[24] Moretti M, Todini A, Baiocchi A, et al. A layered architecture for fair resource allocation in multicellular multicarrier systems. IEEE Transactions on Vehicular Technology, 2011, 60(4): 1788-1798

[25] Zhao C, Topcu U, Li N, et al. Design and stability of load-side primary frequency control in power systems. IEEE Transactions on Automatic Control, 2014, 59(5): 1177-1189

[26] Cai D, Mallada E, Wierman A. Distributed optimization decomposition for joint economic dispatch and frequency regulation//Proceedings of the 54th IEEE Conference on Decision and Control (CDC). Osaka, Japan, 2015: 15-22

[27] Balakrishnan H, Lakshminarayanan K, Ratnasamy S, et al. A layered naming architecture for the Internet. ACM SIGCOMM Computer Communication Review. 2004, 34(4): 343-352

[28] Jones N C, Van Meter R, Fowler A G, et al. Layered architecture for quantum computing. Physical Review X, 2012, 2(3): 031007

[29] El-Samad H, Kurata H, Doyle J C, et al. Surviving heat shock: Control strategies for robustness and performance. Proceedings of the National Academy of Sciences of the United States of America, 2005, 102(8): 2736-2741

[30] Prescott T, Papachristodoulou A. Signal propagation across layered biochemical networks//Proceedings of the 2014 American Control Conference. Portland, USA, 2014: 3399-3404

[31] Shiva S G. Computer Organization, Design, and Architecture. Boca Raton, Florida: CRC Press, 2013

[32] Perry D E, Wolf A L. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992, 17(4): 40-52

[33] Foster I, Kesselman C, Tuecke S. The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 2001, 15(3): 200-222

[34] McCabe T J. A complexity measure. IEEE Transactions on Software Engineering, 1976(4): 308-320

[35] Efatmaneshnik M, Ryan M J. A general framework for measuring system complexity. Complexity, 2016, 21(S1): 533-546

[36] Wallis W D, Shouridge P, Kraetz M, et al. Graph distances using graph union. Pattern Recognition Letters, 2001, 22(6): 701-704

[37] Bunke H. On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters, 1997, 18(8): 689-694

[38] Kelly F P, Maulloo A K, Tan D K H. Rate control for communication networks: Shadow prices, proportional fairness and stability. Journal of the Operational Research Society, 1998, 49(3): 237-252

[39] Taylor R N, Medvidovic N, Dashofy E M. Software Architecture: Foundations, Theory, and Practice. Hoboken, USA: Wiley Publishing, 2009

[40] Zave P, Rexford J. Compositional network mobility// Proceedings of the Verified Software: Theories, Tools, Experiments: 5th International Conference. Menlo Park, USA, 2014: 68-87

[41] Marmsoler D. Towards a theory of architectural styles// Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. Hong Kong, China, 2014: 823-825

[42] Chiang M, Low S H, Calderbank A R, et al. Layering as optimization decomposition: A mathematical theory of network architectures. Proceedings of the IEEE, 2007, 95(1): 255-312

[43] Palomar D P, Chiang M. A tutorial on decomposition methods for network utility maximization. IEEE Journal on Selected Areas in Communications, 2006, 24(8): 1439-1451

[44] Matni N, Doyle J C. A theory of dynamics, control and optimization in layered architectures//Proceedings of the 2016 American Control Conference (ACC). Boston, USA, 2016: 2886-2893

[45] The Grid 2: Blueprint for a New Computing Infrastructure. 2nd Edition. Amsterdam, Netherlands: Elsevier, 2003

[46] Aguiar R L. Some comments on hourglasses. ACM SIGCOMM Computer Communication Review, 2008, 38(5): 69-72

[47] Jacobson V, Smetters D K, Thornton J D, et al. Networking named content//Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. Rome, Italy, 2009: 1-12

[48] Akhshabi S, Dovrolis C. The evolution of layered protocol stacks leads to an hourglass-shaped architecture//Mukherjee A, Choudhury M, Peruani F et al, eds. Dynamics on and of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems. New York, USA: Springer, 2013: 55-88

[49] Popa L, Ghodsi A, Stoica I. HTTP as the narrow waist of the future Internet//Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. Monterey, USA, 2010: 6

[50] Vasilakos A V, Li Z, Simon G, et al. Information centric network: Research challenges and opportunities. Journal of Network and Computer Applications, 2015, 52: 1-10

[51] Zhang L, Afanasyev A, Burke J, et al. Named data networking. ACM SIGCOMM Computer Communication Review, 2014, 44(3): 66-73

[52] Lin Chuang, Jia Zi-Xiao, Meng Kun. Research on adaptive future Internet architecture. Chinese Journal of Computers, 2012, 35(6): 1077-1093(in Chinese)
(林闯, 贾子骁, 孟坤. 自适应的未来网络体系架构. 计算机学报, 2012, 35(6): 1077-1093)

[53] Srivastava V, Motani M. Cross-layer design: A survey and the road ahead. IEEE Communications Magazine, 2005, 43(12): 112-119

[54] Yang M, Li Y, Hu L, et al. Cross-layer software-defined 5G network. Mobile Networks and Applications, 2015, 20(3): 400-409

[55] Barsocchi P, Celandroni N, Davoli F, et al. Radio resource management across multiple protocol layers in satellite networks: A tutorial overview. International Journal of Satellite Communications and Networking, 2005, 23(5): 265-305

[56] Fu F, van der Schaar M. A new systematic framework for autonomous cross-layer optimization. IEEE Transactions on Vehicular Technology, 2009, 58(4): 1887-1903

[57] Miao G, Himayat N, Li Y G, et al. Cross-layer optimization for energy-efficient wireless communications: A survey. Wireless Communications and Mobile Computing, 2009, 9(4): 529-542

[58] Han B, Gopalakrishnan V, Ji L, et al. Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine, 2015, 53(2): 90-97

[59] Tofigh T, Adibi S, Mobasher A, et al. Novel approach to big data collaboration with network operators network function virtualization(NFV). International Journal of Parallel, Emergent and Distributed Systems, 2015, 30(1): 65-78

[60] Carella G, Yamada J, Blum N, et al. Cross-layer service to network orchestration//Proceedings of the 2015 IEEE International Conference on Communications(ICC). London, UK, 2015: 6829-6835

[61] Rehman S, Chen K H, Kriebel F, et al. Cross-layer software dependability on unreliable hardware. IEEE Transactions on Computers, 2016, 65(1): 80-94

[62] Lua E K, Crowcroft J, Pias M, et al. A survey and comparison of peer-to-peer overlay network schemes. IEEE Communications Surveys & Tutorials, 2005, 7(2): 72-93

[63] White B, Lepreau J, Stoller L, et al. An integrated experimental environment for distributed systems and networks. ACM SIGOPS Operating Systems Review, 2002, 36(SI): 255-270

[64] Turner J S, Crowley P, DeHart J, et al. Supercharging planetlab: A high performance, multi-application, overlay network platform. ACM SIGCOMM Computer Communication Review, 2007, 37(4): 85-96

[65] Sherwood R, Gibb G, Yap K K, et al. Can the production network be the testbed?//Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2010). Vancouver, Canada, 2010, 10: 1-6

[66] Sitaraman R K, Kasbekar M, Lichtenstein W, et al. Overlay networks: An akamai perspective. Advanced Content Delivery, Streaming, and Cloud Services, 2014, 51(4): 305-328

[67] Lehman T, Yang X, Ghani N, et al. Multilayer networks: An architecture framework. IEEE Communications Magazine, 2011, 49(5): 122-130

[68] Aberer K, Alima L O, Ghodsi A, et al. The essence of P2P: A reference architecture for overlay networks//Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05). Konstanz, Germany, 2005: 11-20

[69] Navimipour N J, Milani F S. A comprehensive study of the resource discovery techniques in Peer-to-Peer networks. Peer-to-Peer Networking and Applications, 2015, 8(3): 474-492

[70] Wallach D S. A survey of peer-to-peer security issues//Proceedings of the Software Security—Theories and Systems: Mext-NSF-JSPS International Symposium. Tokyo, Japan, 2003: 42-57

[71] Androutsellis-Theotokis S, Spinellis D. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, 2004, 36(4): 335-371

[72] Buyukkaya E, Abdallah M, Simon G. A survey of peer-to-peer overlay approaches for networked virtual environments. Peer-to-Peer Networking and Applications, 2015, 8(2): 276-300

[73] Hu C L, Kuo T H. A hierarchical overlay with cluster-based reputation tree for dynamic peer-to-peer systems. Journal of Network and Computer Applications, 2012, 35(6): 1990-2002

[74] Vu T T, Derbel B, Ali A, et al. Overlay-centric load balancing: Applications to UTS and B&B//Proceedings of the 2012 IEEE International Conference on Cluster Computing. Beijing, China, 2012: 382-390

[75] Kleinberg J. Complex networks and decentralized search algorithms//Proceedings of the International Congress of Mathematicians (ICM). Madrid, Spain, 2006, 3: 1019-1044

[76] Korzun D, Gurtov A. Hierarchical architectures in structured peer-to-peer overlay networks. Peer-to-Peer Networking and Applications, 2014, 7(4): 359-395

[77] Colburn T, Shute G. Abstraction in computer science. Minds and Machines, 2007, 17(2): 169-184

[78] Koponen T, Amidon K, Balland P, et al. Network virtualization in multi-tenant datacenters//Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation. Seattle, USA, 2014: 203-216

[79] Chowdhury N M M K, Boutaba R. A survey of network virtualization. Computer Networks, 2010, 54(5): 862-876

[80] Feamster N, Gao L, Rexford J. How to lease the Internet in your spare time. ACM SIGCOMM Computer Communication Review, 2007, 37(1): 61-64

[81] Chowdhury N M M K, Boutaba R. Network virtualization: State of the art and research challenges. IEEE Communications Magazine, 2009, 47(7): 20-26

[82] Lv B, Wang Z, Huang T, et al. A hierarchical virtual resource management architecture for network virtualization //Proceedings of the 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM). Chengdu, China, 2010: 1-4

[83] Lin Chuang. Performance Evaluation on Computer Networks and Computer System. Beijing: Tsinghua University Press, 2001(in Chinese)
(林闯. 计算机网络和计算机系统的性能评价. 北京：清华大学出版社，2001)

[84] Xue C, Lin C. Performance modelling for transparent computing using stochastic Petri nets. International Journal of Ad Hoc and Ubiquitous Computing, 2016, 21(2): 81-94

[85] Liao Xiang-Ke, Tan Yu-Song, Lu Yu-Tong, et al. On the challenge for supercomputer design in the big data era. Journal of Shanghai University (Natural Science), 2016, 22(1): 3-16 (in Chinese)
(廖湘科，谭郁松，卢宇彤等. 面向大数据应用挑战的超级计算机设计. 上海大学学报：自然科学版，2016, 22(1): 3-16)

[86] Feamster N, Rexford J, Zegura E. The road to SDN: An intellectual history of programmable networks. ACM SIGCOMM Computer Communication Review, 2014, 44(2): 87-98

[87] Hu F, Hao Q, Bao K. A survey on software-defined network and openflow: From concept to implementation. IEEE Communications Surveys & Tutorials, 2014, 16(4): 2181-2206

[88] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[89] Kreutz D, Ramos F M V, Verissimo P E, et al. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, 2015, 103(1): 14-76

[90] Feamster N, Rexford J, Zegura E. The road to SDN. Queue, 2013, 11(12): 20

[91] Tootoonchian A, Gorbunov S, Ganjali Y, et al. On controller performance in software-defined networks//Proceedings of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. San Jose, USA, 2012, 12: 1-6

[92] Tootoonchian A, Ganjali Y. HyperFlow: A distributed control plane for OpenFlow//Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. Berkeley, USA, 2010: 3-3

[93] Koponen T, Casado M, Gude N, et al. Onix: A distributed control platform for large-scale production networks// Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation. Vancouver, Canada, 2010, 10: 1-6

[94] Hu J, Lin C, Li X, et al. Scalability of control planes for software defined networks: Modeling and evaluation// Proceedings of the IEEE 22nd International Symposium of Quality of Service (IWQoS). Hong Kong, China, 2014: 147-152

[95] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing. Communications of the ACM, 2010, 53(4): 50-58

[96] Zhang Q, Cheng L, Boutaba R. Cloud computing: State-of-the-art and research challenges. Journal of Internet Services and Applications, 2010, 1(1): 7-18

[97] Brown A W, Wallnau K C. The current state of CBSE. IEEE Software, 1998, 15(5): 37-46

[98] Banerjee P, Bash C, Friedrich R, et al. Everything as a service: Powering the new information economy. Computer, 2011, 44(3): 36-43

附录 1.

附录 1 对平面、结构、路径和连通分支四个基本概念在本文讨论环境下的定义进行说明，同时对第 5.1 节的中三个引理进行证明.

定义 A1(平面，Plane). 一个平面 P 由一个实体集合 $\{v\}$ 构成，v^{-1} 表示该实体对应的低一层平面中的实体集合. 系统体系结构由物理平面和逻辑平面构成. 任何可发送接收信息的硬件或软件均可以被看作一个实体.

定义 A2(结构，G). 在平面中，关于实体节点集合 $V=\{v\}$ 的结构是指图 $G(V)=G(V,E)$，V 为结构中实体节点集合，$E=\{e(v_j,v_k)\}$ 为结构中连接边集合. $G(P)$ 表示平面 P 的结构.

定义 A3(路径，π). 在结构 $G(V,E)$ 中，若存在有向连接边序列 $\pi(v_j,v_k)=\{v_j,v_1,\cdots,v_k\}$，且 $v_j=v_0$，$v_n=v_k$，$\forall(v_l,v_{l+1})\in E, l\in[0,n-1]$，则称 $\pi(v_j,v_k)$ 为 v_j 到 v_k 的连接路径，连接路径长度为 $d(v_j,v_k)=|\pi(v_j,v_k)|=n$. $d_s=\min\{|\pi(v_j,v_k)|\}$ 为 v_j 到 v_k 的最短连接路径长度，对应

$\pi(v_j,v_k)$称作 v_j到 v_k的最短路径.$d(v_j,v_k)=+\infty$表示 v_j到 v_k不存在连接路径,特别地,$d(v_j,\phi)=+\infty$.

定义 A4(连通分支,CC). 对于 G 的一个子结构 $G'\subseteq G$,如果 $\forall v',v''\in G',v\in G-G',d(v',v'')<+\infty,d(v',v)=+\infty$则称 G'为 G 的一个连通分支,$CC(G)$为 G 所有连通分支构成的集合.特别地,当$|CC(G)|=1$时称 G 为连通结构;对于平面 P,$CC(P)=CC(G(P))$.

引理 1 证明.

由 $\mu(G)$定义知 $\mu(v)=0$,从而,

$$K(v)=\sum_{cc\in CC(v)}[K'(cc)+\sum_{v'\in cc}K((v')^{-1})]$$
$$=K'(v)+K(v^{-1})=K(v^{-1})$$ 证毕.

引理 2 证明.

(1)当 $f_e(n)=n(n-1)/2$时,$\mu(MCS(S,R))=\mu(S)$,记 l'为 G_0中有边($m_j>0$)连通分支数量,则$n_0'=\sum n_j\leq n_0$,$m_0'=\sum m_j=m_0$,$j=1,2,\cdots,l'$,且 $K'(G)=2m[n-m/(n-1)]$关于 n 为增函数.由于没有边的连通分支即为单节点,所以对应结构复杂度为 0.

$$K'(G_0)=2m_0[n_0-m_0/(n_0-1)]$$
$$\geq 2m_0'[n_0'-m_0'/(n_0'-1)]$$
$$=\sum_{l'}2m_j\sum_{l'}[n_j-m_j/(n_0'-1)]$$
$$\geq\sum_{l'}2m_j\sum_{l'}[n_j-m_j/(n_j-1)]$$
$$\geq\sum_{l'}2m_j[n_j-m_j/(n_j-1)]$$
$$=\sum_{l'}K'(G_j)=\sum_{l}K'(G_j).$$

(2)当 $f_e(n)=n-1$时,$\mu(MCS(S,R))=\mu(R)$,记 l'为 G_0中有边($m_j>0$)连通分支数量,则 $n_0'=\sum n_j\leq n_0$,$m_0'=\sum m_j=m_0$,$j=1,2,\cdots,l'$,并且$K'(G)=nm[2-(n-1)/m]=-n^2+(2m+1)n$ 的对称轴为 $n=m+1/2$,由 $m\geq m_r=n-1$知$n\leq m+1\leq 2m$,且有 n 为整数,则知 $K'(G)$在 $n\in[0,m+1]$上为不减函数.由于没有边的连通分支即为单节点,所以对应结构复杂度为 0.

$$K'(G_0)=n_0(2m_0-n_0)+n_0\geq n_0'(2m_0'-n_0')+n_0'$$
$$=\sum_{l'}n_j\left(2\sum_{l'}m_j-\sum_{l'}n_j\right)+\sum_{l'}n_j$$
$$=\sum_{l'}n_j\sum_{l'}(2m_j-n_j)+\sum_{l'}n_j$$
$$\geq\sum_{l'}n_j(2m_j-n_j)+\sum_{l'}n_j$$
$$=\sum_{l'}[n_j(2m_j-n_j)+n_j]$$
$$=\sum_{l'}K'(G_j)=\sum_{l}K'(G_j).$$ 证毕.

引理 3 证明.

(1)当 $f_e(n)=n(n-1)/2$时,$\mu(MCS(S,R))=\mu(S)$,对于边 $m_0\geq m_1+m_2=m_0'$,

$$K'(G)=-\frac{2}{n-1}m^2+2nm,$$

对称轴为 $m=n(n-1)/2$,由 $m\leq m_r=n(n-1)/2$,知 $K'(G)$在 $0\leq m\leq m_r$上关于 m 是增函数,故

$$K'(G_0)-K'(G_1)-K'(G_2)$$
$$\geq K'(G_0')-K'(G_1)-K'(G_2)$$
$$=n_0m_0'\left(2-\frac{m_0'}{m_{r_0}'}\right)-n_1m_1\left(2-\frac{m_1}{m_{r_1}}\right)-n_2m_2\left(2-\frac{m_2}{m_{r_2}}\right)$$
$$=\left(\frac{n_1}{m_{r_1}}-\frac{n_0}{m_{r_0}'}\right)m_1^2+2(n_2-1)m_1-\frac{2m_1m_2n_0}{m_{r_0}'}+$$
$$2(n_1-1)m_2+\left(\frac{n_2}{m_{r_2}}-\frac{n_0}{m_{r_0}'}\right)m_2^2,$$

其中,由于 $n_1=1$时退化为单位映射,不等式取等号成立,当 $n_1\geq 2$时,

$$A=\left(\frac{n_1}{m_{r_1}}-\frac{n_0}{m_{r_0}'}\right)m_1^2+\left(\frac{n_2}{m_{r_2}}-\frac{n_0}{m_{r_0}'}\right)m_2^2$$
$$=2\left(\frac{1}{n_1-1}-\frac{1}{n_0-1}\right)m_1^2+2\left(\frac{1}{n_2-1}-\frac{1}{n_0-1}\right)m_2^2\geq 0,$$

$$B=(n_2-1)m_1-\frac{m_1m_2n_0}{m_{r_0}'}+(n_1-1)m_2$$
$$=m_1\left(n_2-1-\frac{2m_2}{n_0-1}\right)+(n_1-1)m_2$$
$$\geq m_1(n_2-1)\left(1-\frac{n_2}{n_0-1}\right)+(n_1-1)m_2$$
$$\geq m_1(n_2-1)\frac{n_1-2}{n_0-1}+(n_1-1)m_2\geq 0,$$

则 $K'(G_0)-K'(G_1)-K'(G_2)=A+2B\geq 0$.

(2)当 $f_e(n)=n-1$时,$\mu(MCS(S,R))=\mu(R)$,对于边 $m_0\geq m_1+m_2=m_0'$,$K'(G)=2nm-n(n-1)$为关于 m 的增函数,故

$$K'(G_0)-K'(G_1)-K'(G_2)$$
$$\geq K'(G_0')-K'(G_1)-K'(G_2)$$
$$=2(n_0m_0'-n_1m_1-n_2m_2)-n_0(n_0-1)+$$
$$n_1(n_1-1)+n_1(n_1-1)$$
$$=2(n_2-1)[m_1-(n_1-1)]+2(n_1-1)m_2\geq 0.$$ 证毕.

LIN Chuang, born in 1948, Ph.D., professor, Ph.D. supervisor. His research interests include computer networks, performance evaluation, network security analysis, and Petri net theory.

XUE Chao, born in 1988, Ph.D. candidate. His research interests include evaluation and optimization on network architecture, virtual resource scheduling of cloud computing.

HU Jie, born in 1990, Ph.D. candidate. His research interests include performance evaluation and optimization on SDN.

LI Wen-Zhuo, born in 1991, Ph.D. candidate. His research interests include resource management, scheduling and optimization on big data.

Background

Hierarchical design principle has been proposed many years and exists widely in computer architecture design, network architecture design, cloud computing, network virtualization, software engineering and many other branches of computer science. However, few works have been done to explore the essence of principles and methods of hierarchical design, let alone systematic and theoretical insight and analysis. This paper gives a modeling framework for hierarchical design of system architecture, gives the formalized expression of the basic concepts, complexity metrics, performance metrics, principles, mechanisms, roadmaps of hierarchical design. Some insight properties and formal proofs are also been discussed in this paper.

This work is partly supported by the National Natural Science Foundation of China (No. 61472199) and the Tsinghua University Initiative Scientific Research Program (No. 20121087999). These projects aim to provide better principles for the design and modeling evaluation of computer architecture and computing paradigm. Our group has been working on the performance evaluation of the computer networks and computer systems using the stochastic theoretical models, and using optimization techniques to solve the network design problems. Many good papers have been published in respectable international conferences and transactions, such as INFOCOM, IEEE Journal on Selected Areas in Communication and IEEE Transactions on Parallel and Distributed Systems. This paper summarizes the research processes and examples of the hierarchical design principle for computer architecture in a long history and prospect future research challenges.