小 型 微 型 计 算 机 系 统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2025-0243
2026 年 5 月 第 5 期
Vol.47 No.5 2026

数字电网业务场景建模与性能分析方法

单春笑[1,4,5],李永健[2],曾纪钧[3],周兴龙[1,4,5],朱雪阳[1]

[1](中国科学院软件研究所 基础软件与系统重点实验室,北京 100190)
[2](中山供电局,广东 中山 528400)
[3](广东电网有限责任公司信息中心,广州 510308)
[4](国科大杭州高等研究院,杭州 310024)
[5](中国科学院大学,北京 100190)
E-mail:zxy@ios.ac.cn

摘 要:随着数字化转型的深入以及大数据、物联网等技术的兴起,众多新型业务场景不断涌现,相关系统日益复杂.在这一背景下,数字电网领域也面临着全新挑战,尤其是对系统的实时性提出了更为严苛的要求.如何高效地对数字电网业务场景进行性能评估,进而为提高系统的性能可靠性提供支持,是一个重要问题.本文提出了一种面向数字电网业务场景的建模与性能分析方法,并实现了相关原型工具 BizModeler.本文首先针对数字电网领域的特点,提出可视化建模语言 DPG(Dataflow-based Performance Graph);其次总结数字电网领域的通用性能指标,并提出基于同步数据流图的模型性能分析方法;最终通过工具实现以及实例研究验证方法的可用性与有效性.

关 键 词:数字电网;可视化建模;性能分析;同步数据流图
中图分类号:TP393 **文献标识码**:A **文 章 编 号**:1000-1220(2026)05-1271-10

Modeling and Performance Analysis Methods for Digital Power Grid Business Scenarios

SHAN Chunxiao[1,4,5],LI Yongjian[2],ZENG Jijun[3],ZHOU Xinglong[1,4,5],ZHU Xueyang[1]

[1](Key Laboratory of Software and Systems,Institute of Software,Chinese Academy of Sciences,Beijing 100190,China)
[2](Zhongshan Power Supply Bureau,Zhongshan 528400,China)
[3](Information Center Guangdong Power Grid Limited Liability Company,Guangzhou 510308,China)
[4](Hangzhou Institute for Advanced Study,University of Chinese Academy of Sciences,Hangzhou 310024,China)
[5](University of the Chinese Academy of Sciences,Beijing 100190,China)

Abstract:With the deepening of digital transformation and the rise of technologies such as big data and the Internet of Things,many new types of business scenarios are emerging,and the related systems are becoming increasingly complex.In this context,the field of digital power grid is also facing brand-new challenges,especially more stringent requirements on the real-time performance of the system.How to efficiently evaluate the performance of digital power grid business scenarios,and then provide support for improving the performance reliability of the system,is an important issue.In this paper,we propose a modeling and performance analysis method for the digital power grid business scenarios,and implement a prototype tool,BizModeler.We firstly propose a visual modeling language DPG(Dataflow-based Performance Graph)for the characteristics of the digital grid domain;secondly,we summarize the common performance indicators in the digital grid domain and propose a model performance analysis method based on the synchronous data flow graph;finally,we verify the usability and effectiveness of the method through the implementation of the tool as well as an example study.

Keywords:digital power grid;visual modeling;performance analysis;synchronous data flow graph

0 引 言

数字电网应用新一代数字技术如大数据、物联网、人工智能等对传统电网进行改造,旨在使电网更加智能可靠[1].数字电网建设是电网数字化转型的重要途径[2].文献[3-5]中均对数字电网体系架构进行了介绍,在本文中以中国南方电网公司于 2022 年发布的《数字电网标准框架白皮书》[5](以下简称“白皮书”)为准进行研究与讨论.白皮书中提出的数字电网信息系统技术架构包括基础设施层、技术平台层、中台层、应用层等,其中基础设施层涵盖云基础设施、通信网络、边缘计算及终端设备等多个方面,主要负责数据的采集、传输、存储与计算等任务.当前,基础设施层正经历着从云端集中式

收稿日期:2025-06-16 收修改稿日期:2025-08-06 基金项目:广东电网有限责任公司数字电网可信基础设施的建模与分析关键技术项目(037800KC23090002)资助. 作者简介:**单春笑**,女,1999 年生,硕士研究生,CCF 学生会员,研究方向为实时系统性能分析;**李永健**,男,1988 年生,硕士,高级工程师,研究方向为电力网络安全、物联网;**曾纪钧**,男,1987 年生,硕士,高级工程师,研究方向为电力人工智能;**周兴龙**,男,2002 年生,硕士研究生,研究方向为实时系统性能分析;**朱雪阳**(通信作者),女,1971 年生,博士,副研究员,CCF 高级会员,研究方向为形式化方法、实时系统性能分析与调度.

架构向“云管边端”一体化架构的转变,同时也在逐步迈向分布式、功能完善且节能环保的新型基础设施发展道路.在此背景下,对数字电网基础设施业务场景性能的建模与性能分析成为实时提升系统优化、保障系统可靠性以及提升客户使用体验的重要任务.

当前系统建模面临的主要挑战包括模型分析、语言复杂性、建模语言的扩展等方面[6].其中模型分析被认为是最具挑战性的环节.许多建模人员在分析软件模型时遇到困难,这不仅影响了他们对模型的信任度,也降低了模型的应用价值.此外,建模语言的复杂性也是用户普遍关注的问题,特别是那些学习曲线陡峭的语言,使建模人员在学习和使用过程中面临较大阻力.在此背景下,基于数据流的业务流程建模方法[7]为系统建模与分析提供了可行且高效的途径.同步数据流图[8]属于数据流图的重要分支,主要用于实时系统[9,10]的建模与分析验证.同步数据流图建模能够精准地展现系统中的数据流动,尤其是在处理大规模数据流、资源调度和并发任务时,能够为更加精确的性能评估提供支持.

数字电网领域对数据流驱动业务的实时性有较高的要求.基于同步数据流图建模可以直观地呈现业务流程与数据依赖关系,为后续性能分析提供模型依据;数据流建模语言易于理解,使建模人员能够快速上手并投入使用.当前研究较少关注模型与性能分析的紧密集成,尤其在面向数据驱动业务场景的表达能力和建模直观性方面仍存在不足.一些方法虽然具备形式化建模能力,但对实际业务建模者不够友好,缺乏图形化支持,难以满足复杂业务在建模效率与性能分析上的综合需求.此外,现有研究往往未能在工具层面实现建模语言与性能分析的完整支撑,限制了理论成果的应用推广.

针对上述局限,本文所提出的 DPG 建模语言,兼顾了表达能力与直观性.DPG 建模语言结合同步数据流建模理论,专门用于业务流程建模与性能分析.该语言以图形化建模为核心,支持针对不同业务场景的定制化建模.

为了开发更友好的建模语言和工具,需要将场景建模与性能分析技术集成到同一工具中,而这一过程主要面临两大挑战.一方面是现有的同步数据流图工具缺乏良好的可视化建模语言支持,导致建模复杂度高、可视化能力有限,难以直观表达领域特定的业务逻辑.已有工具大多专注于单一功能,要么仅提供场景建模支持,只提供基本的图形化表示,无法直观地展示业务流程和数据流动的细节;要么仅提供性能分析能力,不支持可视化建模分析.

另一个挑战在于如何将业务场景模型与现有的性能分析算法有效结合.业务场景模型通常以高层次的形式描述系统的业务流程和数据流动,而同步数据流模型性能分析算法则需要低层次的数据流图作为输入.这种层次差异导致场景模型与性能分析算法之间需要进行复杂的转换,将高层次的业务场景模型转换为低层次的数据流模型需要大量的工作,且容易引入错误.在转换过程中,业务场景的语义可能会丢失或扭曲,导致性能分析结果不准确.

为应对上述挑战,本文提出了一种数字电网业务场景性能建模与分析方法,并开发原型工具 BizModeler.本文主要贡献点如下:

1)针对数字电网业务场景的特点,提出基于数据流的性能建模语言 DPG(Dataflow-based Performance Graph),并设计该语言的可视化图符表示.该语言能够有效表达业务流程中的数据流动、任务依赖及系统交互.较于传统的流程建模方法,DPG 兼顾直观性与精确性,既方便领域用户使用又足够支持后端算法分析.

2)总结数字电网基础设施业务场景的通用性能指标,提出 DPG 模型性能分析方法.该方法根据不同性能指标将 DPG 模型转换为同步数据流图,而后结合同步数据流图的吞吐量和响应时间分析算法,实现对业务场景的通用性能指标的评估.

3)研制 BizModeler 工具,实现本文提出的建模及分析方法.用户可通过该工具对数字电网业务场景进行建模并分析相关指标.BizModeler 工具支持 DPG 模型管理、可视化编辑和性能分析等功能.

4)通过对数字电网典型业务场景进行实例研究,表明了 BizModeler 能够有效支持数字电网领域的业务场景建模与性能分析.

1 准备知识

本节介绍同步数据流图相关概念,并对本文方法中用到的相关算法进行介绍.

1.1 同步数据流图

同步数据流图 SDFG(Synchronous Data Flow Graph)的概念由 Lee E A 和 Messerschmitt D G 于 1987 年提出[8],旨在解决数字信号处理系统的实时调度与分析问题.

同步数据流图是一种特殊的数据流图,其所有节点的输入输出数据个数是节点定义的一部分.SDFG 是以节点(Actors)和边(Channels)为基本元素的有向图.其中:

节点表示数据处理单元.每个节点在激活时执行一次计算任务,消耗输入数据并产生输出数据.节点的执行次数和输入/输出数据个数在建模时静态确定.本文工作只用到输入输出数据为 1 的情况.

边表示节点之间的数据传输通道,定义节点间的数据依赖关系.边上可能有初始数据(Initial token).

1.2 性能分析算法

本文基于 SDFG 实现业务场景的性能分析,主要关注延迟(Latency)和吞吐量(Throughput)两个指标,接下来介绍两个指标的定义以及所用算法.

延迟定义为从数据到达 SDFG 模型到该数据经过一系列处理后输出的总处理时间:

$$Latency = T_{out} - T_{in} \tag{1}$$

其中 T_{in}是数据进入系统的时间;T_{out}是系统输出数据的时间.

所用算法 $\text{Latency}_{\text{sdfg}}$来源于[11],算法流程概述如下:

1)给定 SDFG G,构造不包含初始数据的边的子图 G_0.

2)对 G_0 进行拓扑排序,确保无环结构,便于后续计算最长路径.

3)依次遍历拓扑排序后的每个顶点 n,计算其延迟量 $\Delta(n)$:

若 n 没有入边,则 $\Delta(n) = n_t$;

若 n 存在入边,则 $\Delta(n) = n_t + Max\Delta(m) \mid m \rightarrow n$,即取所有前驱节点 m 的最大延迟量,并加上自身执行时间.其中 n_t

为节点 n 的执行时间.

4)所求图的延迟为所有顶点的最大延迟量,即:

$$Latency_G = Max_{n \in N} \Delta(n)$$

算法 $Latency_{sdfg}$通过构造子图、拓扑排序和逐节点计算延迟量的方式,求解同步数据流图的延迟时间. 首先,通过移除初始数据边并进行拓扑排序,确保图的无环结构;接着按照拓扑顺序依次计算每个节点的延迟量,通过累加节点执行时间和其前驱节点的最大延迟量,最终得到总延迟时间.

吞吐量为单位时间内系统完成的任务周期数:

$$Throughput = Tasks_{all} / Time_{all} \tag{2}$$

其中 $Tasks_{all}$是任务完成总数,$Time_{all}$是总执行时间.

所用算法 $Throughput_{sdfg}$来源于[12],算法流程概述如下:

1)给定 SDFG G,计算 SDFG 的周期执行向量,并找到其中值最小的节点,记其重复次数为 R_{out}.

2)调度可执行节点,若其入边上的数据量足够,则开始执行;完成节点执行,在出边上输出数据;更新全局时钟,推进最小执行时间,同时记录系统状态,检测是否进入循环.

3)设循环周期内节点执行 R_{out}次,周期时长为 C_{cycle}. 则吞吐为量 R_{out}/C_{cycle}.

4)分解 G 为强连通子图 G_i,计算每个子图的吞吐量 T_i,则 G 的吞吐量为:

$$Throughput_G = Min_i \{T_i \times R_i\} / R_G \tag{3}$$

其中 R_i 表示子图 G_i 的周期向量的最大公约数;R_G 表示 G 的周期向量的最大公约数.

算法 $Throughput_{sdfg}$通过计算 SDFG 的周期执行向量和分析系统状态来评估系统的吞吐量性能. 首先,通过确定最小重复次数的节点和周期时长,计算单个节点的吞吐量;随后,将系统分解为强连通子图,分别计算每个子图的吞吐量,并综合考虑其周期向量的最大公约数,得出整个系统的吞吐量.

2　DPG 建模语言

基于数据流的性能建模语言 DPG 是一种面向业务场景的领域专用建模语言,旨在通过数据流驱动的方式描述系统行为,并支持后续的性能分析.

2.1　DPG 模型定义

定义 1. DPG 模型是一个有向无环图,表示为 DPG = (N, E). 其中:

$N = \{n_1, n_2, \cdots, n_k\}$为节点的集合,每个节点 n_i 表示业务流程中的任务或计算单元.

$E \subseteq N \times N$ 为边的集合,每条边表示任务之间的依赖关系.

每个节点 n_i 表示为一个四元组:

$$n_i = < name, type, p, time >$$

各属性具体含义为:

· *name* 为节点名称,用于唯一标识每个业务任务;

· $type \in T$,表示节点的类型,决定了任务的功能作用;

· $p \in P$,表示节点所在处理器;

· *time* 为节点的处理时间.

其中,T 为节点类型集合,在 2.2 小节中详细说明;P 为可用的处理器集合.

E 中的每个元素(n_i,n_j)是一个有序对,表示从节点 $n_i \sim n_j$的数据流动关系. 若节点 n_i 依赖于节点 n_j的输出,则必须存在一条从 $n_j \sim n_i$ 的边 e_{ij}.

$$E = \{e_{ij} = (n_i, n_j) \mid i = 1, 2, \cdots, m \quad n_i, n_j \in N\}$$

其中,m 是集合 E 中元素的数量.

为了确保 DPG 模型的正确性,需遵循以下约束条件.

DPG 模型的约束条件:

唯一性:每个节点 n_i 和边 e_{ij}在模型中必须具有唯一的标识(名称).

类型:节点类型 type 必须属于预定义的业务任务类型集合 T;处理器 p 必须属于预定义的处理器集合 P.

执行时间:每个节点的执行时间 time 必须为自然数.

结构:DPG 是一个有向无环图,以确保模型无死锁.

部署:部署到同一处理器上的节点必须按照拓扑排序连续,不能出现跨处理器的间隔部署.

2.2　DPG 建模语言的图形化表示

通过对数字电网领域中的业务流程进行梳理,本文总结了常见的 5 种任务类型,这些类型涵盖了不同业务操作的核心功能. 节点类型集合 T 为:

T = {Function, Terminal, Transfer, Vision, Database}

其中,Function 代表功能节点;Terminal 代表终端节点;Transfer 代表传输节点;Vision 代表前端平台节点;Database 代表数据库节点. 节点类型的图符表示及说明见表 1.

表 1　DPG 中不同节点类型的图符表示

Table 1　Graphical representation of different node types of DPG model

类型	图符表示	说明
功能节点 Function		作为业务流程的功能实现模块,负责执行业务逻辑(如数据处理、算法运算).
终端节点 Terminal		代表外部系统的数据输入,如智能配电房中的一次设备与二次设备.
传输节点 Transfer		管理数据在不同任务或系统间的传递,负责数据的传输与转发.
前端平台节点 Vision		代表系统中的可视化平台组件,与用户交互,处理输入指令或输出结果.
数据库节点 Database		表示数据的存储、查询和管理,用来存储系统中的历史数据、配置文件等.

DPG 模型支持为每个节点分配特定的类型,5 种任务类型可灵活组合以构建完整业务流程.

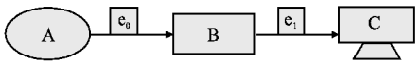

图 1　示例 DPG 模型 M_0 的图形化表示

Fig. 1　Graphical representation of example DPG model of M_0

图 1 展示了 DPG 示例模型 M_0 的图形化表示:首先通过终端节点 A 采集数据,数据通过边 e_0 进入功能节点 B 计算,计算结果通过边 e_1 进入前端平台节点 C 展示. DPG 模型中每个节点 n 的名称(*n. name*)和类型(*n. type*)属性由业务模型本身决定,在建模阶段即已确定,主要用于描述任务的功能

和逻辑结构;而所在处理器($n.\ p$)和执行时间($n.\ time$)则属于部署后才有的属性,它们在业务模型构建时尚未确定.

在明确 DPG 示例模型 M_0 的边以及节点的名称与类型后,接下来需要结合具体的计算资源对模型进行部署,并确定其执行时间属性.

2.3 部署

部署指的是将业务模型中的各个任务(节点)映射到具体的计算资源上. 由于不同处理器的计算能力、并行特性和通信开销不同,同一业务模型在不同的部署方案下可能会呈现出显著的性能差异.

假设任务 n 部署到处理器 p_j 上执行,可以表示为:

$$n.\ p = p_j$$

若任务 n 在处理器 p_j 的执行时间为 t,则表示为:

$$n.\ time = t$$

在业务建模阶段,关注的重点是业务逻辑和功能结构,即前两个属性;然而单独依靠业务建模,无法分析任务在不同处理器资源下的执行性能. 部署之后,节点对应的处理器和执行时间才得以确定. 对于图 1 所示的模型 M_0,它的延迟时间为:

$$L = A.\ time + AtoB + B.\ time + BtoC + C.\ time$$

其中 AtoB 为节点 A 所在处理器到节点 B 所在处理器的通信时间,BtoC 同理. 当存在多种不同硬件设备时,节点在各类硬件上的执行时间与通信时间存在差异,这致使不同的节点部署方案会呈现出不同的延迟时间.

3 DPG 模型性能分析

3.1 方法概述

本文总结了数字电网领域的 6 种常用性能指标,并将其分类为延迟类指标与吞吐量类指标,用于评估系统的性能表现,具体指标介绍将在 3.2 节展开. 其中延迟类指标衡量系统的实时性,对应的性能分析算法为 $\text{Latency}_{\text{sdfg}}$;吞吐量类指标用于评估系统的处理能力,对应的性能分析算法为 $\text{Throughput}_{\text{sdfg}}$.

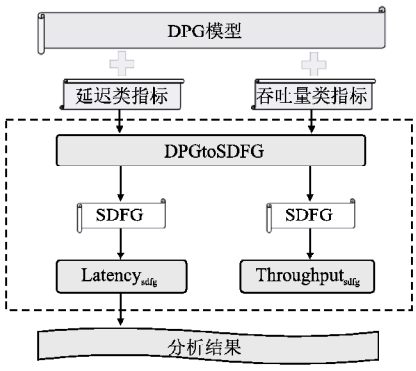

图 2 DPG 模型性能分析方法框架

Fig. 2 DPG framework for the performance analysis of the model

如图 2 所示,用户需要输入 DPG 模型以及待分析的性能指标,性能分析模块(虚线框部分)接收这些输入,并通过转换算法,将 DPG 模型转换为与指定性能指标对应的同步数据流图. 而后调用同步数据流图的性能分析算法,最终输出性能分析结果.

3.2 性能指标

数字电网是智能化的电力系统,涉及电能的实时调度、负荷预测、设备状态监测、故障诊断等关键业务. 在这种场景下,系统性能主要体现在数据交互的实时性、电力调度的吞吐能力、并发任务处理能力等方面. 本文基于数字电网业务场景提出 6 种性能指标:设备日常维护时间、告警信息上报时间、历史消息查询时间、设备查询吞吐量、数据存储吞吐量. 各指标的具体说明见表 2,其中指标类型是指对应于 SDFG 的延迟算法或是吞吐量算法.

表 2 数字电网领域性能指标

Table 2 Performance indicators in the field of digital power grid

指标名称	类型	说明
设备日常维护时间	延迟	设备数据从前端传输到可视化平台的耗时
告警信息上报时间	延迟	终端告警到维护人员接收的响应时间
历史消息查询时间	延迟	数据库查询指定设备历史数据的耗时
设备查询吞吐量	吞吐量	单位时间内设备数据更新操作次数
数据存储吞吐量	吞吐量	单位时间内支持数据写入/读取总量
最大并发告警量	吞吐量	单位时间内可处理的告警数量上限

在发生突发负荷变化时,调度系统必须快速响应,调整负荷分配,以防止电网过载或局部供电中断. 告警信息上报时间直接影响调度中心对异常情况的感知速度;历史消息查询时间关系到电网运行状态的追溯与分析. 智能电网的调度涉及大量分布式节点,调度中心需要协调不同的设备进行负荷调度和电能传输. 为了保证供电稳定性,系统必须在毫秒级别完成数据采集和计算,并在秒级时间内完成调度调整,设备查询吞吐量反映了系统在单位时间内查询电网设备状态的能力,数据存储吞吐量影响系统对海量历史数据的存取效率. 此外,电网运行需要周期性维护,电网调度系统需要对各类传感器采集的数据进行分析,以保障电网的安全、稳定和高效运行,设备日常维护时间也是调度系统优化的重要考量指标,以确保设备稳定运行并降低故障出现的风险. 同时,面对大规模告警事件,系统的最大并发告警量决定了调度中心能够同时处理多少个告警,直接影响电网的安全性与响应速度.

3.3 分析算法

在 3.1 小节中已概述了性能分析方法的基本框架,下面介绍如何将 DPG 模型根据待分析的性能指标转换为适用于现有性能分析算法的 SDFG. 涉及算法的调用关系如图 3 所示.

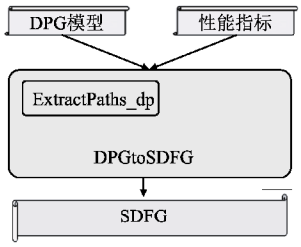

图 3 算法关系图

Fig. 3 Relationship of the algorithms

整个转换流程可以分为两个阶段:

· 数据流路径提取(ExtractPaths_dp):

-从 DPG 模型中筛选符合性能指标的数据流路径,确保所选路径满足业务需求.

-路径搜索 :利用动态规划遍历 DPG 模型,提取所有满足条件的数据流路径.

· 路径合并与 SDFG 构建(DPGtoSDFG):

-对提取出的数据流路径进行合并与转换,生成最终的 SDFG.

-路径整合 :调用算法,对提取的路径进行叠加合并,确保数据流保持正确的拓扑结构.

-SDFG 生成 :根据待分析的性能指标,对合并后的路径图进行转换,使其适用于已有的性能分析算法.

为了构建符合性能指标要求的 SDFG,需要首先在 DPG 模型中提取相应的数据流路径再转换. 在本文的性能分析方法中,路径抽取是关键的一步,其主要作用包括:

1)为性能评估提供精确的分析对象. 不同的执行路径可能有不同的业务逻辑、计算开销和通信延迟,通过路径抽取,可以识别所有与待分析性能指标有关的执行路径,从而深入分析路径上的性能情况.

2)支持 DPG 到 SDFG 转换. 如果没有正确的业务数据流路径抽取,就无法构造合理的 SDFG,导致后续分析不准确.

在路径提取算法中,本研究针对不同的性能指标,要求用户在建模时必须包含特定的关键节点. 以数字电网基础设施业务场景中的设备日常维护时间指标为例,规定待提取的 DPG 路径中必须包含终端节点 n_T、数据库节点 n_D 和可视化平台节点 n_V. 这一要求源于维护流程的核心步骤:设备数据采集、查询设备信息、可视化反馈. 具体来说:

1)终端节点负责实时提供设备状态数据,是维护工作的起点;数据库节点负责存储和查询维护历史,确保维护任务具有参考依据;可视化平台节点负责 展示维护状态和进度,确保维护人员能及时响应. 缺少任何一个节点,都会导致流程数据不完整.

2)设备维护时间 = 数据采集时间 + 查询处理时间 + 响应执行时间,这 3 个节点正对应了设备维护的输入、计算、输出关键环节. 因此,确保路径包含终端节点 n_T、数据库节点 n_D 和可视化平台节点 n_V,可以保证数据流的完整性,使性能分析更具实际价值.

3)在 DPG 转换为 SDFG 过程中,若路径不包含这些节点,转换后的数据流图可能丢失关键业务环节,影响建模的准确性.

本文提出基于动态规划(Dynamic Programming,DP)的路径提取方法. 该方法依赖于 n_T、n_D、n_V 3 个关键节点,提取 DPG 模型中从起始节点 n_T 到终止节点 n_V,且经过节点 n_D 的所有路径. 算法 ExtractPaths_dp 输入 DPG 模型,返回所有符合条件的路径集合.

算法 ExtractPaths_dp 结合动态规划存储已计算路径,减少重复计算. 该算法基于拓扑排序和动态规划. 首先,算法检查 DPG 模型是否包含 3 个关键节点,若缺失则直接返回错误. 然后,对图进行拓扑排序,以确保按照依赖顺序遍历各个节点,并初始化 dp 用于存储路径列表,用于存储从 n_T 节点到当前节点的所有可能路径. 算法流程如图 4 所示.

接着,算法 ExtractPaths_dp 从 n_T 节点开始,将其作为初始路径存入 dp. 按照拓扑排序顺序,依次处理每个节点,并遍历其所有出边,将可达的下一节点 next 加入当前路径,并存入 dp[next]. 在处理 n_V 节点时,算法停止拓展,转而检查 dp[n_V]中的路径,筛选出包含 n_D 节点的路径并存入 resultPaths,最终返回所有符合条件的路径集合.

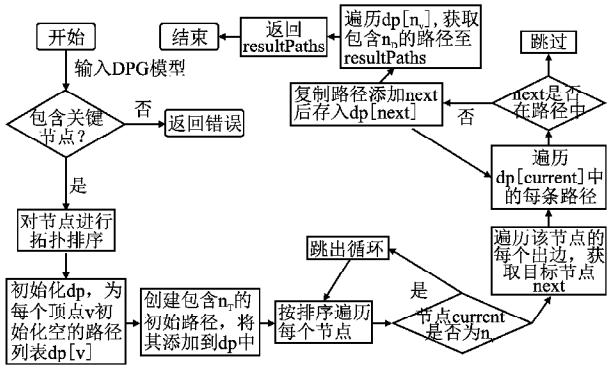

图 4　算法 ExtractPaths_dp

Fig. 4　Algorithm ExtractPaths_dp

算法 DPGtoSDFG 根据路径提取算法后得到 DPG 模型中所有符合目标条件的路径,而后根据待分析指标的不同类型转换得到 SDFG 图. 具体算法流程如图 5 所示. 首先,调用算法 ExtractPaths_dp 提取所有符合指标要求的数据流路径. 然后,初始化一个空的 SDFG 结构 sdfG,并遍历所有路径,将路径上的节点和边逐步添加到 SDFG 中.

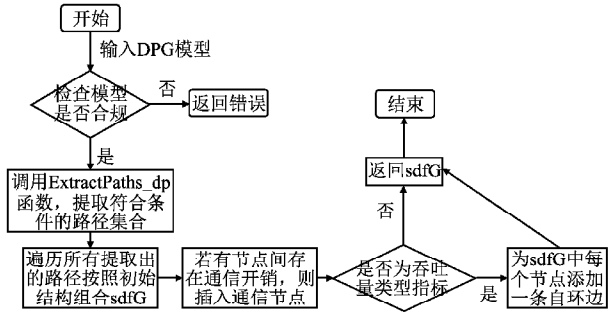

图 5　算法 DPGtoSDFG

Fig. 5　Algorithm DPGtoSDFG

对于每条路径,算法 DPGtoSDFG 按照顺序遍历路径中的每个节点 current,若该节点尚未加入 sdfG,则将其添加进去. 随后,检查 current 的下一个节点 next,若 next 不在 sdfG 中,则添加 next,并检查 DPG 中是否存在 current → next 的边,若该边存在且 sdfG 尚未包含该边,则将其加入 sdfG.

在 DPGtoSDFG 转换过程中,将所有符合条件的数据流路径合并之后,还需要考虑按照待分析指标的类型进行结构转换,不同的性能分析需求决定了对 SDFG 结构的转换方式.

对于需要调用延迟时间分析算法 Latency_{sdfg} 的性能指标,当两个节点 n_s、n_t 运行在不同的计算单元上时,由于不同处理器之间的数据传输可能带来额外的通信开销,算法会在二者之间插入一个额外的通信节点,并添加相应的传输边,以模拟跨处理器通信的时间开销. 插入通信节点为 n_c:

$$n_c = (name_c, Transfer, null, time_c)$$

其中：

--$name_c$ 表示通信节点的名称，

--*Transfer* 代表该节点的类型，表示数据传输过程，

--*null* 说明该节点不需要在处理器上执行，仅用于数据传输，

--$time_c$ 表示该通信节点的执行时间，其值取决于两个节点 n_s 和 n_t 所在的处理器之间的通信时间.

插入通信节点 n_c 后需添加边：

$$e_{sc}=(n_s,n_c),e_{ct}=(n_c,n_t)$$

对于需要调用吞吐量分析算法 $\text{Throughput}_{\text{sdfg}}$ 的性能指标，为了限制 SDFG 中的并发度，除了添加通信节点与对应边，还需要为每个计算节点 n_i 添加自环边 e_{ii}：

$$e_{ii}=(n_i,n_i)$$

4 实例研究

本文基于上述 DPG 建模与性能分析方法，实现了原型工具 BizModeler. 以下简述 BizModeler 架构及实例研究.

4.1 工具实现

BizModeler 工具提供了 DPG 模型的可视化建模与性能分析支持，使用户能够在工具中构建业务场景的 DPG 模型，并直接进行性能分析.

iDFOS[13] 是一支持 SDFG 分析、调度与优化的工具. 本文工作所需调用的 SDFG 性能分析算法 $\text{Latency}_{\text{sdfg}}$ 和 $\text{Throughput}_{\text{sdfg}}$ 均来自 iDFOS. $\text{Latency}_{\text{sdfg}}$ 主要用于计算数据流从输入到输出的总时延，评估系统的响应时间，而 $\text{Throughput}_{\text{sdfg}}$ 用于计算单位时间内可处理的最大任务数量，衡量系统的并发能力. 这些算法在 iDFOS 工具中已验证其准确性和可靠性.

BizModeler 采用模块化设计，总体架构如图 6 所示，包括前端建模交互层、模型管理层、性能分析层 3 大部分，这 3 个模块相互协作，实现从业务建模到性能分析的完整流程.

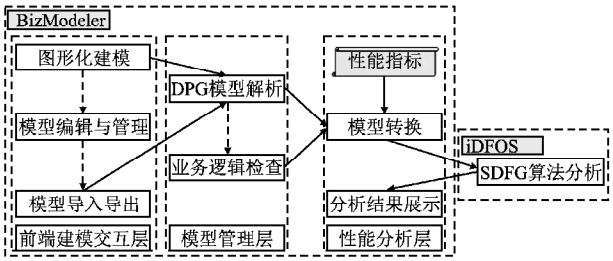

图 6 BizModeler 层次架构图

Fig. 6 Hierarchical diagram of BizModeler

BizModeler 提供基于 DPG 的拖拽式建模界面，支持用户直观构建业务模型. 通过拖拽式交互，用户可以直观地构建 DPG 模型，包括添加、删除和修改业务节点. 在完成建模后，用户可以对 DPG 模型进行编辑和管理；在进行性能分析时，系统将 DPG 模型按照不同的性能指标需求转换为 SDFG，该过程通过 DPGtoSDFG 算法实现. 而后基于 SDFG 进行吞吐量、响应时间的分析，这一部分算法在 iDFOS 中实现.

· 前端建模交互层是 BizModeler 的用户界面部分，提供可视化建模功能，使用户能够直观地构建业务模型. 该模块主要包含 图形化建模、模型编辑与管理、模型导入与导出 3 大功能. 前端建模交互层采用 Java Swing 框架进行开发，结合 JGraph 组件实现图形化业务建模. 同时，利用 JTree 组件实现模型层次结构的可视化，并通过事件监听处理用户的节点选择与编辑操作. 此外，采用自定义渲染来优化树形结构的显示效果，并结合 UndoableEditEvent 实现建模操作的撤销与恢复功能.

· 模型管理层是负责解析用户建模的业务流程，并确保其结构和逻辑的正确性. 该层主要包含 DPG 模型解析、业务逻辑检查两个子模块. 模型管理层主要负责对 DPG 模型进行解析、存储及一致性检查，采用 Java 数据结构（如 HashMap、Set、List 等）进行节点和边的管理，并基于 DefaultTreeModel 维护层次结构. 此外，在业务逻辑检查方面，采用规则校验机制，确保模型符合约束条件，如节点的合法连接关系、循环依赖检测等，避免构建无效模型.

· 性能分析层的主要功能是对业务模型进行性能评估，并提供可视化分析结果. 该层包含 性能指标、模型转换、SDFG 算法分析、分析结果展示 4 个部分. 性能分析层负责根据待分析的性能指标将 DPG 转换为 SDFG，并调用 iDFOS 预先实现的性能分析算法. 转换部分采用动态规划提取业务数据流路径，并构建 SDFG 结构. 在延迟分析时，会插入通信节点模拟异构处理器间的传输延迟；在吞吐量分析时，会额外添加节点自环边以控制节点并发. 将转换后的 SDFG 输入 iDFOS 预先实现的性能分析算法，获取分析结果. 最终，分析结果通过前端界面进行展示.

BizModeler 界面主要包括主界面（Main）、节点属性输入界面（Detail）和分析结果展示界面（Analysis）. 主界面用于新建/导入、编辑、分析、导出模型；节点属性输入界面用于更改对应节点的名称、类型、部署方案等属性；分析结果展示面用于展示模型分析结果. Analysis 界面直接展示性能分析结果，无需用户操作；节点属性输入界面，用于业务建模过程中节点属性的详细配置，确保用户可以根据具体业务需求，自定义任务名称、类型、处理器信息以及计算时间. 主界面主要由菜单栏、组件模板、模型分析、硬件配置、模型编辑 5 部分组成. 其中：

· 菜单栏由“文件”、“退出”、“编辑”、“说明”4 个部分组成，提供了 BizModeler 的一些常用功能按钮.

· 组件模板栏提供不同类型节点模板，用户只需单击相应的节点形状，系统会自动在模型中插入一个新的节点，并允许用户进一步配置其属性.

· 模型分析栏列出可分析的性能指标，用户可以对所构建的 DPG 模型进行性能分析，系统会根据用户选择的性能指标，自动调用相应的 SDFG 性能分析算法.

· 硬件配置栏包括“硬件平台信息”、“节点执行时间”、“部署方案”3 个页面. 3 个部分的文件可以分别导入导出，也可以作为一整个项目存取. 在“硬件平台信息”页面用户可以自定义预设的处理器类型、个数以及不同类型处理器间通信时间；在“节点执行时间”页面用户可以定义模型节点在不同处理器类型上的执行时间；“部署方案”页面用于设置模型节点部署在哪个处理器上.

· 模型编辑栏提供可视化编辑窗口供用户编辑 DPG

模型.

4.2　实例介绍

数字电网基础设施涉及海量的实时数据采集、传输、处理以及多层次的决策支持,任何一个环节的延迟或故障都可能引发连锁反应,影响整个电网的稳定运行. 例如,电网实时数据来自与各种传感器和监控设备的电力系统状态数据[14],在设备故障发生时,如果告警数据不能在一定时间内上报平台并及时定位故障点采取措施,可能导致大面积停电. 因此,对数字电网基础设施流程进行合理建模与性能分析是一件十分必要且有效的技术保障措施.

下面以图 7 所示的数字电网基础设施业务场景 GridFlow 为例,展示 DPG 建模方法及分析结果.

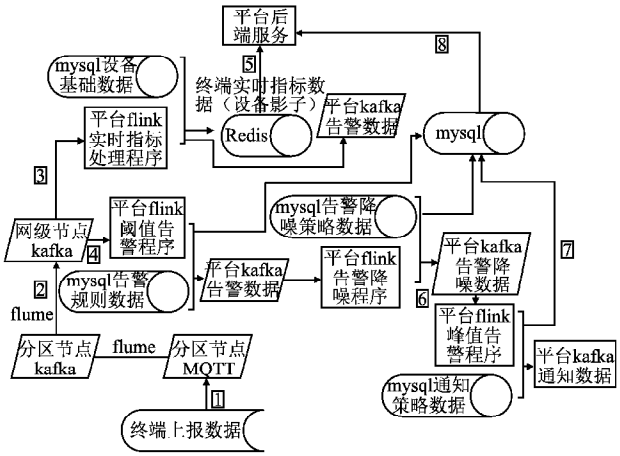

图 7　数字电网基础设施业务场景 GridFlow

Fig. 7　Digital grid infrastructure business scenario GridFlow

从图 7 中可以总结出以下流程:

1)终端设备广泛分布于电网各环节(如发电端传感器、输电线路监测装置、用户端智能电表等),持续采集海量数据.

2)这些数据通过高速网络传输至实时数仓,实时数仓承担着数据存储、整理与初步处理的重任,需要具备强大的数据吞吐能力与快速的数据处理速度,以确保数据的及时性与准确性.

3)在传输过程中,MySQL 与 Redis 数据库按需保存临时数据或长期数据,并且允许存取有关的设备信息以及重要的历史数据信息.

4)可视化平台将数字电网中的复杂数据以直观易懂的形式呈现给运维人员、管理人员以及决策层,辅助他们进行电网状态监测、故障诊断以及战略决策等工作.

在明确该业务场景的数据流程后,可以对其进行业务建模,将每个具体任务映射为 DPG 模型 GridFlow_DPG 中的节点,并为其指定一个唯一的名称,并选定节点的类型. 然后,根据任务间的数据流动关系,使用有向边将相关节点连接起来,从而构建出该场景的初步业务模型. 图 8 为 DPG 模型 GridFlow_DPG 的图形化表示.

在完成 DPG 业务模型的初步构建后,已确定每个节点的 name、type 属性以及模型中的边,在部署之前,每个节点的 p 及 time 属性为空.

接下来需要结合具体的计算资源对模型进行部署,即将各个节点映射到不同的处理器,并确定其执行时间属性. 由于

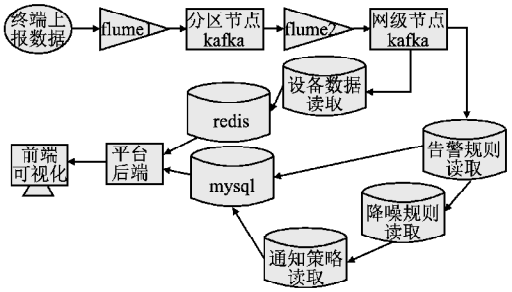

图 8　GridFlow 的 DPG 模型 GridFlow_DPG

Fig. 8　DPG model GridFlow_DPG of GridFlow

不同处理器的计算能力、并行特性和通信开销不同,同一业务模型在不同的部署方案下可能会呈现出显著的性能差异.

4.3　实验设置

本文实验在 12 核 2.60 GHz CPU、32GB DDR5 内存、操作系统为 Windows 11 的电脑上运行.

定义可用于部署的处理器类型有 3 种:$Proc_0$、$Proc_1$ 和 $Proc_2$. 处理器间的通信时间设置如表 3 所示. 同构处理器之间的传输时间较短,而异构处理器之间的传输时间较长. 模型 GridFlow_DPG 各节点在不同处理器类型上的执行时间如表 4 所示.

表 3　处理器间的通信时间

Table 3　Communication time between processors

通信时间	$Proc_0$	$Proc_1$	$Proc_2$
$Proc_0$	10	20	30
$Proc_1$	20	10	50
$Proc_2$	30	50	10

表 4　GridFlow_DPG 的节点在各类处理器上的执行时间

Table 4　Execution time of GridFlow_DPG's nodes on different processor types

	$Proc_0$	$Proc_1$	$Proc_2$
终端上报数据	25	20	20
flume1	10	15	15
分区节点 kafka	30	30	30
flume2	15	20	20
网级节点 kafka	25	20	30
设备数据读取	25	20	30
redis	35	35	25
告警规则读取	30	35	30
降噪规则读取	25	15	2
通知策略读取	20	25	25
mysql	15	20	35
平台后端	55	35	60

本文为模型 GridFlow_DPG 设置了 3 种部署方案. 前两种方案假设资源充足,即处理器数量多于模型中的节点数,使得每个节点可以独占一个处理器.

第 1 种方案采用同构处理器,所有节点均运行在相同类型的处理器上;第 2 种方案采用异构处理器,即节点部署在不同类型的处理器上. 由于所研究的场景均涉及云-边-端协同

部署的业务流程,在实际环境中,每个任务节点通常拥有专属的计算资源.因此,设置这两种节点独占处理器的部署方案是合理的;第 3 种方案则考虑资源受限的情况,即处理器数量少于模型中的节点数,此种方案假设系统共有 3 个处理器.

表 5 部署方案设置

Table 5 Deployment schemes

	部署方案 1		部署方案 2		部署方案 3	
节点名称	n. p	n. time	n. p	n. time	n. p	n. time
终端上报数据	$proc_2^1$	20	$proc_0^1$	25	$proc_0^1$	25
flumel	$proc_2^2$	15	$proc_0^2$	10	$proc_0^1$	10
分区节点 kafka	$proc_2^3$	30	$proc_0^3$	30	$proc_0^1$	30
flumc2	$proc_2^4$	20	$proc_0^4$	15	$proc_0^1$	15
网级节点 kafka	$proc_2^5$	30	$proc_0^5$	25	$proc_0^1$	25
设备数据读取	$proc_2^6$	30	$proc_2^1$	30	$proc_1^1$	20
redis	$proc_2^7$	25	$proc_2^2$	25	$proc_1^1$	35
告警规则读取	$proc_2^8$	30	$proc_2^3$	30	$proc_1^1$	35
降噪规则读取	$proc_2^9$	20	$proc_2^4$	20	$proc_1^1$	15
通知策略读取	$proc_2^{10}$	20	$proc_2^5$	20	$proc_1^1$	25
mysql	$proc_2^{11}$	25	$proc_2^6$	25	$proc_1^1$	20
平台后端	$proc_2^{12}$	35	$proc_1^1$	35	$proc_2^1$	35
前端可视化	$proc_2^{13}$	60	$proc_1^2$	45	$proc_2^1$	60

表 5 展示了不同部署的方案.其中 n. p 表示节点部署的处理器,n. time 表示节点的执行时间.总结不同方案的处理器需求如下:

部署方案 1:需要 13 个 proc2 处理器.

部署方案 2:需要 5 个 $proc_0$ 处理器,2 个 $proc_1$ 处理器,6 个 $proc_2$ 处理器.

部署方案 3:需要 1 个 $proc_0$ 处理器,1 个 $proc_1$ 处理器,1 个 $proc_2$ 处理器.

4.4 实验结果

表 6 展示了不同部署方案下性能分析的结果以及执行时间.性能指标包括设备日常维护时间、告警消息上报时间、历史消息查询时间、设备查询吞吐量、最大并发告警量、数据存储吞吐量.

执行时间为在 BizModeler 工具中进行操作的实际时间,以毫秒为单位,分为 3 部分:“转换时间”指将 DPG 模型转换为 SDFG 所需的时间,实例研究中采用 3.3 中的 DPGtoSDFG 算法进行模型转换;“性能分析时间”指执行性能分析算法所需的时间;“总时间”则指用户从点击待分析指标到界面展示结果所需的总时长.

实验表明,资源受限情况对系统吞吐量有较大影响.例如,方案 3 的吞吐量类性能指标(设备查询吞吐量、最大并发告警量、数据存储吞吐量)的分析结果明显低于方案 1 和方案 2,表明处理器资源受限的部署方案在系统进行高吞吐操作时存在性能瓶颈.相比之下,方案 2 的吞吐量类指标分析结果普遍较高,说明在混合部署($proc_0$、$proc_1$、$proc_2$ 结合使用)的情况下,系统在吞吐性能上更均衡.此外,在该方案下处理器之间的通信时长对吞吐量类指标影响较大.

处理器资源对于延迟类指标(如设备日常维护时间、告警信息上报时间、历史消息查询时间)无明显影响.然而,部署方案 3 中的通信开销最低,一定程度上降低了延迟类指标的分析结果.这表明在资源受限的情况下,优化通信开销可以有效改善延迟类指标的性能表现.

表 6 GridFlow_DPG 的实验结果

Table 6 Experimental results of GridFlow_DPG

	性能指标	分析结果	执行时间(毫秒)		
			转换	性分析	总时间
方案 1	设备日常维护时间	345	3.53	1.47	78.49
	告警信息上报时间	405	4.05	1.64	78.23
	历史消息查询时间	140	1.93	2.36	79.20
	设备查询吞吐量	1/60	3.51	14.73	95.12
	最大并发告警量	1/60	2.55	2.44	90.65
	数据存储吞吐量	1/60	1.76	3.52	85.68
方案 2	设备日常维护时间	380	3.35	2.35	76.44
	告警信息上报时间	440	3.55	1.74	88.24
	历史消息查询时间	165	2.85	0.92	76.71
	设备查询吞吐量	1/50	2.47	12.78	87.60
	最大并发告警量	1/50	3.45	12.18	88.22
	数据存储吞吐量	1/50	1.36	3.72	81.89
方案 3	设备日常维护时间	325	2.49	0.47	78.44
	告警信息上报时间	365	3.95	0.66	81.55
	历史消息查询时间	165	3.01	0.69	79.43
	设备查询吞吐量	1/105	3.12	13.50	96.35
	最大并发告警量	1/105	4.49	18.01	102.71
	数据存储吞吐量	1/95	2.76	6.28	85.01

通过分析不同部署方案在资源受限情况下的性能表现,以及对吞吐量和延迟等关键性能指标的影响机制,可以为实际部署提供更具针对性的优化建议.在资源受限的情况下,优先选择混合部署方案(如方案 2),通过合理分配处理器资源,避免单一处理器的瓶颈,从而提高系统的吞吐量.

在延迟类指标的优化中,应重点关注通信开销的降低.例如方案 3 通过减少通信开销显著改善了延迟类指标的性能表现.对于延迟敏感的应用,建议优化数据传输路径和通信协议,以进一步降低通信延迟.

在实际部署中,建议根据具体应用场景的需求,灵活选择部署方案.对于需要高吞吐量的场景,优先考虑混合部署;对于延迟敏感的场景,优先考虑优化通信开销.

5 相关工作

本文工作涉及建模与性能分析,下面介绍相关工作.

5.1 建模语言

从应用范围角度来看,建模语言可以分为通用建模语言(GPML,General-Purpose Modeling Language)和领域专用建模语言(DSML,Domain-Specific Modeling Language)两大类[15,16].GPML 具有广泛适用性,能够用于多种领域和场景的建模;而 DSML 针对特定领域或问题设计,具有高度的专业性和针对性.下面主要介绍 DSML.

DSML 也表示为领域专用语言(Domain Specific Language,DSL)[17].领域专用建模语言针对特定领域设计,其设计目标是提供一种简洁、高效的建模方式,以满足特定场景的建模需求,帮助从业人员解决与其特定领域相关的复杂建模问题[18].针对各领域的特定建模语言种类繁多,主要包括应

用于业务流程建模的业务流程模型与符号(Business Process Modeling Notation,BPMN)[19],应用于嵌入式系统和数字信号处理的同步数据流图(Synchronous Dataflow Graph,SDFG)[9,10,20],以及应用于智能应用[21]、物联网[22]等各领域的专用建模语言.

数据流图建模专注于数据流动的细节,属于领域专用的建模方法.同步数据流图是数据流图的一种特定形式,是一种用于表示实时并行数据处理系统的图形模型.SDFG 更强调数据流动的同步性和时间特性,通常用于实时系统建模,便于分析和优化并行计算.

领域专用建模语言的符号表示可以采用图形、文本或二者结合的方式[23].图形化 DSML 采用图形元素,如形状、线条、箭头、图标等,来直观地表示建模概念、关系和结构.文本 DSML 则使用文本形式的语法、词汇和语句来描述建模元素及其关系,通过对关键字、标识符、运算符等的组合来精确表达模型的各个方面.

当前研究较少关注模型与性能分析的紧密集成,尤其在面向数据驱动业务场景的表达能力和建模直观性方面仍存在不足.一些方法虽然具备形式化建模能力,但对实际业务建模者不够友好,缺乏图形化支持,难以满足复杂业务在建模效率与性能分析上的综合需求.此外,现有研究往往未能在工具层面实现建模语言与性能分析的完整支撑,限制了理论成果的应用推广.

本文所提出的 DPG 建模语言,兼顾了表达能力与直观性.DPG 建模语言结合同步数据流建模理论,专门用于业务流程建模与性能分析.该语言以图形化建模为核心,支持针对不同业务场景的定制化建模.在技术实现上,所开发的 BizModeler 工具进一步支撑并验证了该建模语言的实用性与有效性,满足复杂业务场景下的直观建模和性能分析需求.

5.2　性能分析方法

性能分析是业务场景建模的重要目标,旨在评估系统的吞吐量、延迟、资源利用率等关键指标.本节重点关注基于 SDFG 模型的系统性能分析.针对 SDFG 的性能分析方法主要包括静态调度与周期分析、吞吐量优化与流水线设计、形式化验证等,这些方法可在系统设计早期阶段提供精确的性能评估.

Sriram 等人[24]通过将 SDFG 转换为 HSDFG 进行性能分析,首先,需要检查 SDFG 规范的一致性,并计算其非零迭代向量.接着,应用算法将 SDFG 转换为 HSDFG,并通过两阶段流程将其映射到目标平台:第 1 阶段确定 HSDFG 节点在处理器上的分配;第 2 阶段为每个处理器生成静态顺序调度.总体目标是在平台资源约束下最大化吞吐量.然而,由于吞吐量受到任务分配和调度策略的影响,而映射和调度的组合数量呈指数级增长,使得搜索最优解变得极具挑战性.

一些工作[25-27]采用展开和重定时技术,以实现最佳迭代周期并优化 HSDF 调度.重定时是指通过调整 SDFG 边上的初始标记分布,优化数据流动路径,减少关键路径延迟,从而提高系统吞吐量.展开是指通过对 SDFG 进行多次迭代展开,增加任务的并行度,使更多计算任务能够同时执行,进一步提高吞吐量.

SDFG 因其在静态调度、周期分析、吞吐量优化等方面的良好分析能力,已在嵌入式系统和实时流处理系统中得到广泛应用.然而,这些方法多集中于算法流程、嵌入式应用或硬件系统,对于业务场景的适配性与建模直观性仍显不足.

本文在现有 SDFG 性能分析研究的基础上,通过将 DPG 模型转换为 SDFG,并集成现有的 SDFG 性能分析算法,以量化评估业务场景的关键性能指标.将 DFG 模型转换为同步数据流图后,能够有效利用 SDFG 分析算法支持性能分析和验证,这一转换流程既保持了 DPG 模型的直观表达能力,又结合了 SDFG 的强大分析能力.

5.3　工具对比

本文也开展了工具 BizModeler 与类似工具的综合对比,涵盖建模可视化、性能分析以及开发语言等方面.支持同步数据流图性能分析的工具种类繁多,其中 DIF[28]、Ptolemy II[29]、SDF3[30]和 iDFOS[13]是 4 个较为代表性的工具.本文对比了 BizModeler 与其他几种相关工具.如表 7 所示,这些工具各自针对不同的应用场景和建模需求,具备不同的优缺点.

表 7　工具对比

Table 7　Comparison of tools

工具名称	主要用途	可视化建模	性能分析	开发语言	建模语言
BizModeler	业务场景性能建模与分析	√	√	Java	DPG
DIF	数字信号处理领域分析处理	×	√	Java	DIF
Ptolemy II	可视化建模仿真与系统设计	√	×	Java	多种(包括 SDFG)
SDF3	SDFG 优化与调度分析	×	√	C++	SDFG
iDFOS	SDFG 调度优化与性能分析	√	√	Java	SDFG

BizModeler 主要面向业务场景的性能建模与分析,能够直观地表示业务流程和数据流,支持可视化建模和性能评估.DIF 主要用于数字信号处理领域的数据流建模,尽管具备强大的分析能力,但缺乏可视化支持.Ptolemy II 以系统建模与仿真为核心,但未提供专门的性能分析功能,且其使用门槛较高.

在 SDFG 建模与优化方面,SDF3 是一个专门的工具,适用于嵌入式系统和实时系统的性能优化.然而它主要关注 SDFG 优化与调度,缺乏直观的可视化界面,可能影响用户的建模效率.而 iDFOS 作为一款 SDFG 可视化建模与分析工具,同时支持调度优化,适用于通用 SDFG 的研究与分析,但未针对特定业务场景进行定制.

从对比结果来看,BizModeler 结合了建模与性能分析能力,专门用于数据流驱动的业务场景.在数据流驱动的业务场景建模与分析方面,BizModeler 具备显著优势,兼顾可视化建模与性能评估,与现有工具形成互补.

6　总结与展望

本文针对数字电网业务场景,提出了 DPG 建模语言以及 DPG 模型性能分析方法,并实现了相关原型工具 BizModeler,对典型业务场景的实例研究表明本文工作的有效性.在将来的工作中,将支持定义更多领域相关的性能指标;同时根据

实际应用情况提升工具的易用性,优化界面设计,以满足更多用户需求.

References:

[1] China southern power grid company limited[R]. Digital Power Grid White Paper,2020.

[2] Huang Y,Liu J,Zhang Y,et al. The connotation,characteristics and development of digitized distribution network:a review[C]//IEEE 6th Conference on Energy Internet and Energy System Integration,2022:2690-2693.

[3] Bai H,Wang Y. Digital power grid based on digital twin:definition,structure and key technologies[J]. Energy Reports,2022,8(16):390-397.

[4] Yan S. Feasibility and utilization of information integration technology in digital power grid[C]//International Conference on Internet of Things,Robotics and Distributed Computing(ICIRDC),2023:427-432.

[5] China southern power grid company limited[R]. Digital Power Grid Standard Framework White Paper,2022.

[6] Ozkaya M,Erata F. Understanding practitioners' challenges on software modeling:a survey[J]. Journal of Computer Languages,2020,58:100963,doi:10.1016/j.cola.2020.100963.

[7] Dolean C C,Petrusel R. Data-flow modeling:a survey of issues and approaches[J]. Informatica Economica,2012,16(4):117-128.

[8] Lee E A,Messerschmitt D G. Synchronous data flow[J]. Proceedings of the IEEE,1987,75(9):1235-1245.

[9] Singh A,Ekberg P,Baruah S. Applying real-time scheduling theory to the synchronous data f low model of computation[C]//29th Euromicro Conference on Real-Time Systems(ECRTS),2017:8:1-8:22.

[10] Khatib J,Munier Kordon A,Klikpo E C,et al. Computing latency of a real-time system mod eled by synchronous dataflow graph [C]//Proceedings of the 24th International Conference on Real-Time Networks and Systems,2016:87-96.

[11] Leiserson C E,Saxe J B. Retiming synchronous circuitry[J]. Algorithmica,1991,6(1):5-35.

[12] Ghamarian A H,Geilen M C,Stuijk S,et al. Throughput analysis of synchronous data flow graphs[C]//6th International Conference on Application of Concurrency to System Design(ACSD),2006:25-36.

[13] Zhu X Y. idfos:a tool for dataflow[EB/OL]. http://lcs.ios.ac.cn/~zxy/tools/idfos.html,2014.

[14] Li Z,Qiu Z,Yang S,et al. Research on the application of digital twinning in digital trans formation of power grid[C]//3rd International Conference on Artificial Intelligence and Autonomous Robot Systems(AIARS),2024:34-39.

[15] Fedeli A,Beutling N,Laurenzi E,et al. Comparison of general-purpose and domain-specific modeling languages in the iot domain:a case study from the omilab community[M]. Aachen:Sun SITE,Informatik V,RWTH Aachen,2023:145-157.

[16] Bogale B,Vesinurm M,Lillrank P,et al. Visual modeling languages in patient pathways:scoping review[J]. Interactive Journal of Medical Research,2024,13(1):e55865.

[17] Fishwick P A. Handbook of dynamic system modeling[M]. Boca Raton:CRC Press,2007.

[18] Gupta R,Jansen N,Regnat N,et al. User-centric model-aware recommendations for indus trial domain-specific modelling languages [C]//ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C),2023:330-341.

[19] Farshidi S,Kwantes I B,Jansen S. Business process modeling language selection for research modelers[J]. Software and Systems Modeling,2024,23(1):137-162.

[20] Marrone P,D'Angelo S,Fontana F,et al. Ciaramella:a synchronous data flow programming language for audio dsp[C]//Proceedings of the 19th Sound and Music Computing Conference,2022:412-419.

[21] Laurenzi E,Ruggli O,Van der Merwe A. Bpmn4mopla:mobility planning based on business decision-making[M]. Springer,Cham,Switzerland,2022:617-638.

[22] Corradini F,Fedeli A,Fornari F,et al. X-IoT:a model-driven approach for cross-platform IoT applications development[C]//Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing,2022:1448-1451.

[23] Predoaia I. Towards systematic engineering of hybrid graphical-textual domain-specific languages [C]//ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion(MODELS-C),2023:153-158.

[24] Sriram S,Bhattacharyya S S. Embedded multiprocessors:scheduling and synchronization[M]. Boca Raton:CRC Press,2018.

[25] O'Neil T W,Sha E H M. Combiningextendedretiming and unfolding for rate-optimal graph transformation[J]. Journal of VLSI Signal Processing Systems for Signal,Image and Video Technology,2005,39(3):273-293.

[26] Zhu X Y. Efficient retiming of unfolded synchronous dataflow graphs[C]//24th International Conference on Engineering of Complex Computer Systems(ICECCS),2019:134-143.

[27] Zhu X Y,Geilen M,Basten T,et al. Multiconstraint static scheduling of synchronous dataflow graphs via retiming and unfolding [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,2015,35(6):905-918.

[28] Hsu C J,Keceli F,Ko M Y,et al. Dif:an interchange format for dataflow-based design tools[EB/OL]. http://dspcad.umd.edu/dif/,2004.

[29] Brooks C,Lee E A,Liu X,et al. Ptolemy project:heterogeneous modeling and design[EB/OL]. https://ptolemy.berkeley.edu/ptolemyII/index.htm,2008.

[30] Stuijk S,Geilen M,Basten T. SDF3:SDF for free[EB/OL]. https://www.es.ele.tue.nl/sdf3/,2006.

附中文参考文献:

[1] 中国南方电网有限责任公司[R]. 数字电网白皮书,2020.

[5] 中国南方电网有限责任公司[R]. 数字电网标准框架白皮书,2022.