计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(4):130−136 [doi: 10.15888/j.cnki.csa.008453]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 SVG 和 Vue 的数据过程可视化设计①

林　晨, 邓　录, 董　璐

(中国电子科技集团公司第五十二研究所, 杭州 310012)
通信作者: 林　晨, E-mail: lincc343@163.com

摘　要: 针对传统数据过程系统无法动态地显示数据过程及对数据过程中的算子无法进行拖拽编辑的问题, 本文提出基于 SVG 和 Vue 的数据过程可视化方案并给出实现过程. 该方案基于 SVG 高扩展性及不依赖于分辨率和 Vue 数据双向绑定等优点, 可以对数据过程进行拖拽式创建或者编辑, 给用户带来动态的交互体验, 并且该方案具有高扩展性, 还能提高开发效率和保证良好性能.

关键词: SVG; Vue; 数据双向绑定; 扩展性; 动态; 可视化分析; 大数据

引用格式: 林晨,邓录,董璐.基于 SVG 和 Vue 的数据过程可视化设计.计算机系统应用,2022,31(4):130–136. http://www.c-s-a.org.cn/1003-3254/8453.html

Design of Data Progress Visualization Based on SVG and Vue

LIN Chen, DENG Lu, DONG Lu

(The 52nd Research Institute, CETHIK Group Co. Ltd., Hangzhou 310012, China)

Abstract: Given that the traditional data process system cannot display the data process dynamically and drag and edit the operators in the data process, this study proposes a data process visualization scheme based on scalable vector graphics (SVG) and Vue and gives the implementation process. With the high scalability and independence from resolution of SVG and the two-way data binding of Vue, the scheme can drag and drop the data process to create or edit it and thereby bring a dynamic interaction experience to users. The scheme, with high scalability, can also improve development efficiency and deliver favorable performance.

Key words: SVG; Vue; two-way data binding; scalability; dynamic; visualized analysis; big data

1　引言

大数据时代, 数据成爆炸式增长, 并且数据类型多样, 格式复杂, 但是存在数据分散, 数据间关联少的问题, 无法进一步对数据深度利用. 数据过程就是对数据的深度利用和对大数据的处理, 包含数据采集, 录入, 标准化转换和输出的整个过程. 数据过程可视化是为了使用户能更清晰地了解数据深度利用和大数据处理的过程, 形象地显示数据过程, 加深理解大数据处理. 但是传统的数据过程的可视化只是显示整个数据过程, 不能对数据过程进行拖拽式编辑, 无法对特定业务需求的算子进行添加和数据录入. 现有的可视化解决方案主要有导入 Visio 流程图、百度的 ECharts 库、蚂蚁金服的 AntV G6 库和美国专家开发的 D3 库.

现有的可视化方案存在一定的缺陷, 比如使用 Visio 绘制流程图, 功能强大, 但是只能以静态图片形式展示. 基于 Web 技术开发的可视化图表库, 如 ECharts 库, AntV G6 库和 D3 库. ECharts 库适用于基础图表, 对于用户交互需求较强不适用. AntV G6 提供了一系列设计优雅、便于使用的图可视化解决方案, 能帮助开发者搭建属于自己的图可视化、图分析、或图编辑

① 收稿时间: 2021-06-11; 修改时间: 2021-07-07, 2021-07-20, 2021-08-13; 采用时间: 2021-08-24; csa 在线出版时间: 2022-03-22

器应用[1]. 但是 AntV G6 可视化解决方案扩展性差, 对于模块间的连线规则或者相互关系有较多需求的流程图不适用. D3 (data-driven documents)[2] 是基于 SVG 和 JQuery 开发的数据驱动文档. D3 将强大的可视化、动态交互和基于数据驱动的 DOM 操作完美结合一起, 直接操作真实的 DOM 节点, 将数据转化为可视化图形[3]. D3 与 AntV G6 相比扩展性更高, 不依赖于分辨率, 但是对于大数据情形, 如果数据经常发生改变, 图表就会不断地重新渲染, 即会经常操作真实 DOM, 耗费性能.

针对上述存在的问题, 本文提出基于 SVG 和 Vue 技术的数据过程可视化设计和实现方案. 该方案可以动态、直观地显示整个数据过程, 能对数据过程中的算子进行拖拽式创建和编辑, 丰富用户的交互体验. 该方案扩展性强, 不依赖于分辨率, 采用虚拟 DOM 技术渲染页面[4], 不会频繁操作真实 DOM, 对数据进行双向绑定, 只需要考虑数据的变化, 不需要考虑页面元素的改变, 提高了开发效率和保证了良好性能.

2 关键技术

2.1 Vue

Vue.js 是以数据驱动为核心的用户交互式的渐进式 Web 框架[5], 与其他前端框架的差异是, Vue 的核心构建思想是采取自底向上的设计, 使用数据双向绑定和组件化构建单页面应用[6]. Vue 最大的优势是使用虚拟 DOM 来渲染页面, 不需要频繁操作真实 DOM, 数据双向绑定, 提高了性能.

2.2 RESTful Web

RESTful Web[7] 服务是基于 HTTP 应用协议的标准化、通用化的操作实现对资源的操作[8]. 客户端通过 URL, 使用不同的 HTTP 请求方法来获取和处理资源, 并且响应可以被标志为可缓存或者不可缓存, 将客户端和服务端上的资源分离, 提高了网络效率[9].

2.3 Element UI

Element UI 是饿了么公司开发的基于 Vue 的开源组件库, 包含常见的组件如按钮, 输入框等, 可快速实现页面开发[10].

3 系统设计

3.1 系统框架设计

数据过程可视化系统提供动态添加及连接不同的算子来对大数据深度利用和处理的功能. 基于 B/S 架构开发的数据过程系统框架如图 1 所示.

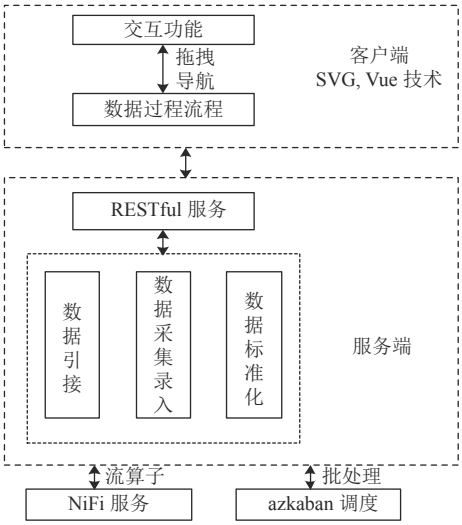

图 1 系统框架图

B/S 架构分为客户端和服务端两部分, 数据过程服务端是通过 RESTful 接口接收数据, 根据数据的类型 (批、流), 流算子通过 NiFi 服务执行, 批处理通过 azkaban 调度, 将任务封装好交给 azkaban 执行. 客户端是基于 Vue、ElementUI 框架和 SVG 技术构建的数据过程流程图页面. SVG 技术实现算子模型和算子间的连接线的创建, 并基于 Vue 的数据双向绑定来实现算子在面板上的移动、导航、编辑及删除等操作功能. 本系统数据库的设计是用户自主选择数据库, 包括达梦数据库, MySQL 和 Oracle. 客户端和服务端的通信使用 Ajax 异步请求进行 JSON 数据的传输, 同时采用 HTTPS 协议保证传输过程的安全性和保密性. 本文重点阐述数据过程客户端的功能设计和实现过程.

3.2 客户端功能设计

系统的客户端功能包含算子面板, 画布, 工具栏, 导航和属性 5 个模块, 功能设计图如图 2 所示.

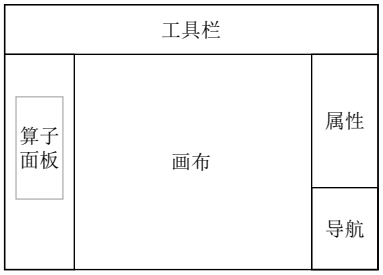

图 2 数据过程客户端设计图

算子面板模块负责显示处理数据深度利用的算子, 比如采集算子和处理算子, 并提供算子的拖拽式创建功能. 画布模块负责大数据处理过程的创建和处理, 通过采集算子获取数据库目录下的某一类数据, 连接处理算子对该数据进行去重或者异常值剔除等功能. 工具栏模块负责算子的复制、删除和画布位置缩放的操作功能. 导航模块负责快速定位画布中算子的位置. 属性模块负责显示算子的属性, 并将修改后的属性值存储到数据库, 若属性数据不符合规则, 则不能执行整个数据过程. 本文重点分析画布模块和属性模块的设计.

3.3 系统各模块详细设计

3.3.1 画布模块

画布模块负责大数据处理过程的创建和处理, 包含算子绘制和算子连接功能. 本系统是基于采集算子和处理算子对复杂大数据进行深度利用和处理, 采集算子是指数据源类的算子, 比如关系型数据库采集算子; 处理算子是连接采集算子后对数据进行处理的一类算子, 如异常值剔除算子, 算子绘制是基于 SVG 技术实现. 通过采集算子获取达数据库目录下的某一类数据, 可以连接多种处理算子对该类数据处理, 比如数据清洗和数据替换. 不同算子的连接是基于 SVG 技术绘制连接线, 连接规则是基于 Vue 技术构建, 比如处理算子必须前置连接采集算子, 一个算子如果超过最大连接限制数, 其他算子则不能连接该算子. 算子相互连接完成后执行数据过程, 这样会将处理后的数据存放到指定数据表中, 画布模块的原理如图 3 所示.

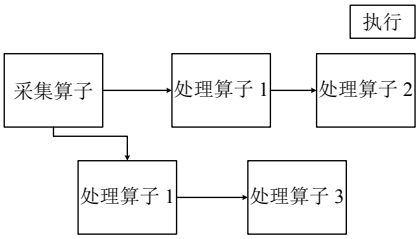

图 3 画布模块原理图

3.3.2 属性模块

属性模块为算子的主要配置属性, 算子的连接和使用都依赖于属性数据. 基于 Vue 技术将绘制的算子和从数据库中获取的算子属性数据双向绑定, 修改算子的配置属性存储到数据库, 不需要重复修改数据结构, 动态地更新算子. 如果不按照规则修改算子属性, 则整个数据过程无效. 算子的属性数据量大, 为了简化数据结构和降低数据传输率, 将属性对象转化为 JSON 字符串. 采集算子和处理算子主要属性信息表如表 1 和表 2 所示.

表 1 采集算子主要属性表

database	datalog	table	key	field	fieldType
数据库	数据目录	数据表	主键	字段名	字段类型

表 2 处理算子主要属性表

field	fieldType	method	treaty	value	partition
字段名	字段类型	处理方式	协议	字段值	分区字段

4 系统实现

本章重点讲述基于 SVG 和 Vue 技术的数据过程实现. 该系统的实现基于 Vue 组件化机制分为算子, 画布, 属性, 导航和工具栏 5 大组件. 基于 SVG 的绘制和 Vue 的数据双向绑定功能, 数据过程可视化系统的实现方案主要包括绘制功能, 数据构建及传递和交互功能.

4.1 绘制功能

绘制一个数据过程包含算子绘制和算子连接两大要素. 算子连接是绘制有逻辑关系的两个算子之间的线段, 并且需要满足相应的连接规则[11].

4.1.1 算子绘制

算子面板和画布组件需要对算子进行绘制. 算子面板组件是基于 Element UI 的 Collapse 折叠面板实现算子的分类, 算子分为采集组件和处理组件两大类, 如图 4 所示.

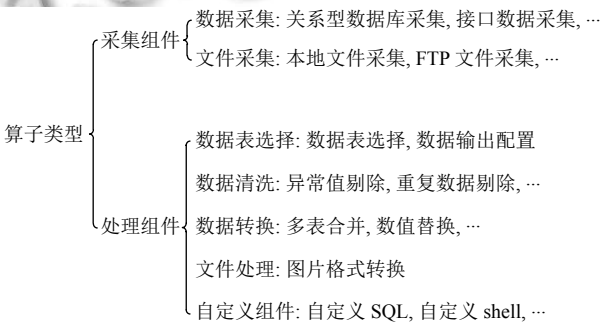

图 4 算子类型

算子绘制是基于 SVG 二维矢量图的一个绘制空间, 首先使用<svg></svg>标签构建空间, 再使用分组元素 g 来表示一组算子. 每个算子使用矩形元素<rect></rect>绘制, 设置 x、y 的属性值表示算子的坐标, width 和 height 的属性表示算子的长宽, 再设置 rx 和 ry 属性使

矩形产生圆角. 算子的图片是使用 pattern 元素定义, 通过属性 type 对应, 然后在 rect 标签中使用 fill 属性引用来填充, 算子创建过程的流程图如图 5 所示.

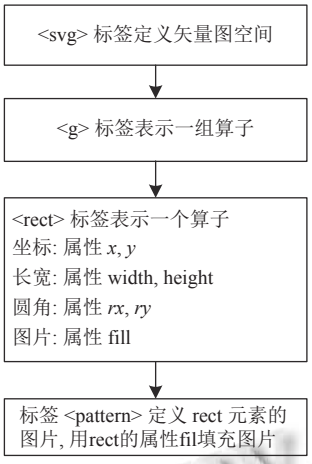

图 5 算子创建过程

画布组件中的算子是数据过程的一个重要元素, 绘制的算子需要包含 4 个连接点来连接其他算子. 选择的算子会显示 4 个连接点, 不被选中的算子则不显示. 算子的连接点是使用<circle>标签创建, 4 个点的相对坐标分别为 (−0.5, 0), (0.5, 0), (0, −0.5), (0, 0.5), 再通过属性 transform 向 X 轴平移算子的 X 轴坐标乘以点的相对 X 轴坐标, 向 Y 轴平移算子的 Y 轴坐标乘以点的相对 Y 轴坐标, 这样算子的 4 个连接点就分别位于算子 4 条边的中心位置, 如图 6 所示.

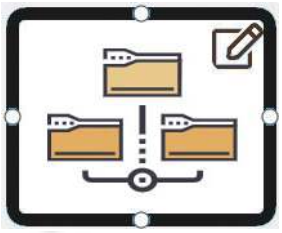

图 6 算子

算子数据包括唯一 ID, 名称, 类型, 长宽, 类型, X 轴和 Y 轴坐标, 4 个连接点相对坐标所构成的数组, 属性 JSON 字符串, 连接规则数组, 参数和描述信息. 由于算子属性对象中的属性比较多, 为了简化数据结构和降低数据传输率, 将属性对象转化为 JSON 字符串. 基于 Vue 的数据双向绑定, 算子数据贯穿于整个数据过程.

4.1.2 算子连接

算子的连接算法是根据算子间的连接规则进行连接. 连接线数组存放连接线唯一的 ID, 连接线对应的源算子和目标算子的唯一 ID, 数据, 连接点位置. 连接规则数组是存放算子的逻辑关系以及最大, 最小连接数.

两个算子连接, 首先判断两算子是否相同, 然后判断目标是否存在连线规则, 如果没有规则直接连接, 否则根据目标算子的连线规则中类型是否等于或者包含源算子的类型得到过滤后的连接规则数组, 再判断连接数量是否超出规定的最大或者最小连接数, 未超过则两个算子完成连接, 实现流程如图 7 所示. 两个不同的算子连接如果不满足规则不能连接, 即设置连接点的透明度为 0, 不显示连接点, 如果满足连接规则, 设置连接点的透明度为 1.

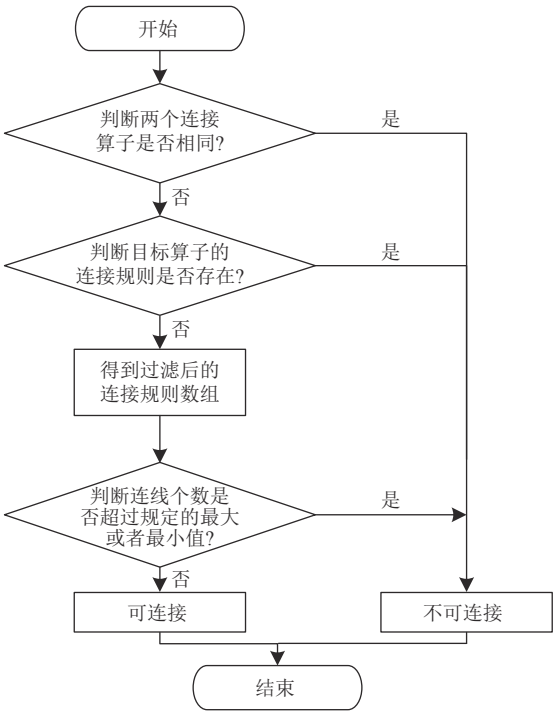

图 7 算子连接规则流程图

元素 path 的属性 *d* 表示路径数据, 即表示满足连接规则的源算子和目标算子的路径数据, *d* 的值是一个“命令+参数”的序列, 表现形式类似为 (*M*10 10 *L*20 20 40 50), *M* 表示移动到点坐标, *L* 表示连线到点坐标. 算子连接需要明确是源算子的哪个连接点和目标算子的哪个连接点连接, 并且计算连接点在画布上的 X 轴和 Y 轴的坐标, 计算公式为源算子连接点 X 轴坐标等于源算子连接点的相对坐标乘以宽度加上源算子的 X 轴

坐标, Y 轴和目标算子连接点坐标计算方法类似. 为了避免连线被算子遮挡, 连接线会出现一个拐点, 该拐点的坐标就是源算子连接点和目标算子连接点的 X 轴坐标的最大值, Y 轴最大值, 如果源算子和目标算子在同一条直线上则拐点因为坐标相同显示直线, 如图 8 所示.

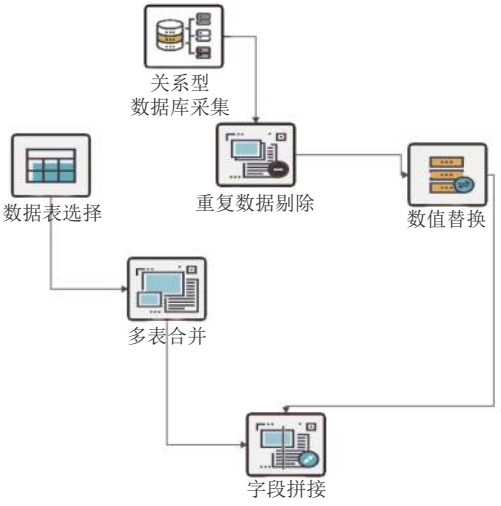

图 8 数据过程连接线

源算子指向目标算子的连线, 使用箭头区分源算子和目标算子. 箭头采用 SVG 的元素 polygon 绘制, 属性 points 定义三角形每个角的 X 轴和 Y 轴坐标. 箭头的坐标为目标算子连接点的坐标, 将直角坐标转化为极坐标, 计算连接点的角度, 并旋转对应的角度得到箭头指向角度, 效果图如图 8 所示.

4.2 数据构建及传递

Vue 基于 MVVM (model-view-ViewModel)[12] 架构, 在 View 层的操作数据传递给 ViewModel 层, Model 层随之更新, ViewModel 层会根据 Model 层的数据变化自动更新, 重新渲染[13]. 本系统基于 Vue 实现, 数据部分由 JavaScript 实现, 不需要专门处理视图层的 HTML 节点, 视图的变化随着数据变化, 提高了性能和开发效率. 比如实现数据过程需要构建连线数组 edgelist 和算子数组 itemlist, 视图模版核心代码如下所示:

```
<g v-for="d in edgelist">
  <path :d="d"  class="for-click" ></path>
  <polygon points=" " transform=""/>
</g>
<itemnode v-for="i in itemlist"></itemnode>
```

算子数组 itemlist 用来显示在画布上的算子, 算子的所有属性数据都在 itemlist 数组中. Edgelist 数组对象包含自身的 id, 源算子 id 和属性 from 对象, 目标算子 id 和属性 to 对象, 其中 from 对象和 to 对象分别包含源算子和目标算子的所有属性, 坐标以及连接点的相对位置坐标, 数据结构如图 9 所示. <path>连线算法是根据 edgelist 数组中源算子连接点的坐标和目标算子连接点的坐标进行连线绘制.

```
{
    direction: '',  id:'',  sourceId: '',  destinationId: '',
    from: {item: {},  index: ''},
    to: {item:{},  index:  ''}
}
```

图 9 Edgelist 数据结构

本文系统的实现基于 Vue 组件化机制, 分为算子, 画布, 属性, 导航和工具栏 5 大组件. 每个组件实例的作用域是独立的, 父组件的数据需要通过 prop 传递给子组件, 子组件需要通过 emit 方法将数据传递给父组件. 设置每个组件的唯一的 ref 值, 非父子关系的组件间通过调用组件的数据或者方法获得, 比如组件 buttonbar 为工具栏组件, 组件 viewport 为画布组件, 分别设置组件 ref 值, 工具栏组件通过调用画布组件得到连接线 edgelist 数组, 核心代码示例如下:

```
<buttonbar ref="buttonbar"
:edgelist="$refs.viewport.edgelist">
</buttonbar>
<viewport ref="viewport"></viewport>
```

本文算子组件, 画布组件, 属性组件, 导航组件和工具栏组件为非父子组件, 各组件数据传递如图 10 所示.

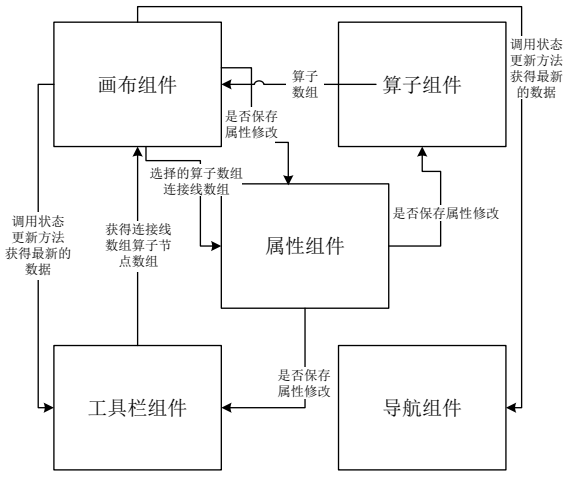

图 10 各组件数据传递图

4.3 交互功能

为了提升用户的交互体验, 数据过程的交互功能主要包括拖拽移动、放大或者缩小算子和导航. 基于 SVG 实现的数据过程, 可以自由缩放, 并不会影响图片的失真. 本系统通过鼠标的滚轮 mousewheel 事件实现放大或者缩小功能, 并设置整个画布缩放的最大和最小的值. 通过鼠标的 mouseup 事件, mousedown 事件和 mousemove 事件计算得到算子最终移动的坐标位置来实现算子的拖拽移动. SVG 具有高扩展性, 可直接使用 Vue 的 v-html 指令将画布标签<svg>的所有数据显示在页面上, 并设置成一定的缩放比例, 使用鼠标 mousemove 事件移动位置来实现导航功能, 效果图如图 11 所示.

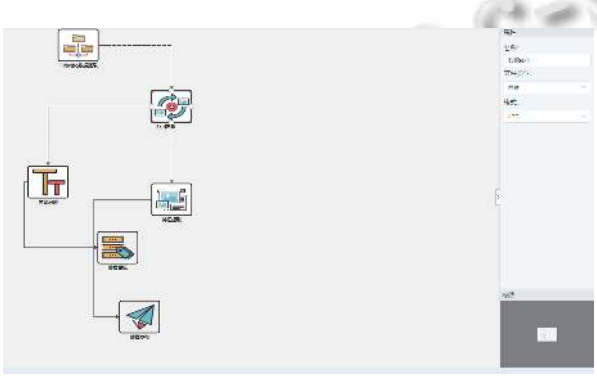

图 11 导航效果图

5 应用效果评估

5.1 应用平台和数据集

本系统采用的硬件环境配置为 Intel x86 处理器, 128 GB 内存, 软件运行平台为 CentOS 7.2 操作系统. 系统 Web 客户端运行平台为 64 位 Windows 7 操作系统, Chrome 81.0 浏览器. 本系统验证实验采用了内部不公开的数据集. 该数据集包含 200 个算子, 每个算子包含至少 10 个属性字段, 配置属性至少含有 20 个属性字段. 每个算子有 4 个连接点, 每个连接点可以连接其他符合规则的算子.

5.2 效果评估

为了验证系统方案的有效性, 分别从分辨率, 响应帧数 FPS, 扩展性, 交互性, 开发效率及功能完善度和现有方案做对比.

根据表 3 看出, 本系统方案和 D3 库不依赖于分辨率, 任意缩放不会失真, 而其他 3 个方案都依赖于分辨率, 任意缩放会失真. 因为本方案和 D3 库使用 SVG 绘制的矢量图; G6 库, ECharts 方案底层是 Canvas 渲染的位图. 矢量图不依赖于分辨率, 任意缩放不失真, 而位图依赖于分辨率, 任意缩放会失真. 数据过程的性能主要通过算子拖拽, 移动, 连线操作的响应时间来体现. 使用 Chrome 浏览器提供的 performance 工具中的 FPS (每一秒的帧数) 指标来测试对比本方案和现有方案的页面动画的性能指标. 通过表 2, 从 FPS 维度可以看出本方案, G6 库和 D3 库在相同时间间隔的 FPS 达到 60 以上, 说明未出现卡顿, 用户体验良好.

表 3 数据过程系统性能对比表

系统 维度	分辨率	FPS	扩展性	交互性	开发效率	功能完 善程度
本方案	不依赖	65	高	高	高	高
导入Visio图	依赖	无	低	无	无	无
G6库	依赖	68	低	高	中	中
D3库	不依赖	62	高	高	低	高
ECharts库	依赖	56	低	低	中	中

本系统方案和现有方案从多个维度的对比, 从扩展性维度看出 G6 库和 ECharts 库因是商业产品不能二次开发扩展性差, 导入 Visio 图方案不能对实时的算子数据进行修改扩展性差, D3 库和本系统方案可以定制化开发, 更具有高扩展性. 从交互性维度得出, 除了导入 Visio 图方案和 ECharts 库外其他方案都有较高的交互性. 从开发效率维度看, G6 库每一次数据的更新都需要重新组装数据, D3 库采用 JQuery 库开发代码复杂且效率低, 本系统方案基于 Vue 库的组件化和数据双向绑定特点开发, 开发效率高. 从功能完善程度对比得出 G6 库和 ECharts 因是商业产品不能进行二次开发, 功能有限, D3 库和本系统方案可以自主开发功能, 功能完善度更高更容易把控. 综合来看本系统方案在分辨率, 响应帧数, 扩展性, 交互性, 开发效率和功能完善程度多个方面对比其他现有方案更具有优势, 且能保证良好性能和丰富的用户体验.

6 总结

本文针对传统数据过程系统无法动态地显示数据过程及对数据过程中的算子无法进行拖拽编辑的问题, 提出了一种基于 SVG 和 Vue 的数据过程可视化系统方案. 该系统采用 SVG 绘制元素, 可以任意缩放, 不依赖于分辨率, 不会造成图片像素的失真, 并且 SVG 可

直接嵌套部分 Element UI 组件库使用, 还可以直接使用 Vue 的方法指令, 具有扩展性强和高开发效率的特点. 基于 Vue 的组件化和数据双向绑定特点, 本系统方案实现不需要关注视图 HTML 节点的处理, 只需要处理数据部分, 采用虚拟 DOM 来渲染页面, 不需要频繁操作真实 DOM, 提高了开发效率同时保证了良好性能.

参考文献

1 AntV 蚂蚁数据可视化. https://antv.vision/zh.

2 邱秀连, 康倩, 王峥. 人物关系的可视化研究. 计算机系统应用, 2018, 27(4): 27–33. [doi: 10.15888/j.cnki.csa.006294]

3 范伟梅. 基于 D3 的数据可视化图表系统 [硕士学位论文]. 广州: 华南理工大学, 2017.

4 刘翔宇. 基于 Vue 的数据可视化系统的设计与实现 [硕士学位论文]. 北京: 北京邮电大学, 2018.

5 杨加玉. 基于检索词扩展和文本表示的文库搜索引擎 [硕士学位论文]. 西安: 长安大学, 2017.

6 吴志霞, 叶根梅, 甘丽, 等. 基于 Vue. js 框架实现移动终端数据可视化研究与实践. 通化师范学院学报, 2020, 41(6): 62–66.

7 章武媚. 基于 RESTful Web 技术的资源管理系统设计与实现. 计算机应用与软件, 2014, 31(5): 23–28. .

8 李俊杰. 基于 RESTful Web 服务的配电设备状态监测系统研究与开发 [硕士学位论文]. 南宁: 广西大学, 2019.

9 袁赟. Java 与 Restful Web Service. 电脑知识与技术, 2007, 4(21): 780–782.

10 吴昌政. 基于前后端分离技术的 web 开发框架设计 [硕士学位论文]. 南京: 南京邮电大学, 2020.

11 朱建军. 基于 D3 的可视化组件开发及其在研讨系统中的应用 [硕士学位论文]. 武汉: 湖北工业大学, 2015.

12 王志任. 基于 Vue. js 的开发平台的设计与实现 [硕士学位论文]. 广州: 广东工业大学, 2018.

13 黄佛辉. 基于 Vue. js 的 WebGIS 开发研究 [硕士学位论文]. 重庆: 重庆交通大学, 2017.