(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 105808237 A
(43)申请公布日 2016.07.27

(21)申请号 201610105312.8

(22)申请日 2016.02.25

(71)申请人 北京京东尚科信息技术有限公司
地址 100080 北京市海淀区杏石口路65号西杉创意园西区11C楼东段1-4层西段1-4层
申请人 北京京东世纪贸易有限公司

(72)发明人 赵振阳

(74)专利代理机构 北京成创同维知识产权代理有限公司 11449
代理人 蔡纯 张靖琳

(51)Int.Cl.
G06F 9/44(2006.01)

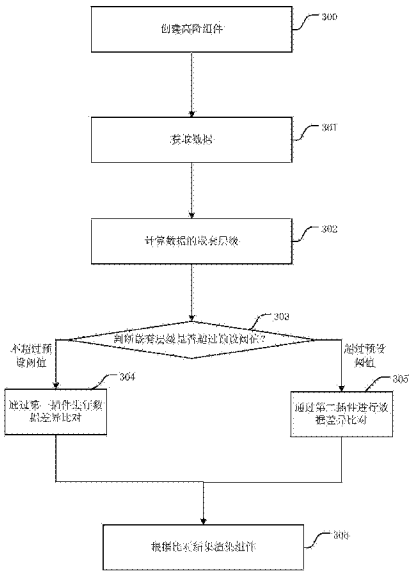
权利要求书1页 说明书6页 附图4页

(54)发明名称

页面渲染方法和页面渲染系统

(57)摘要

本实施例提供的页面渲染方法，包括创建高阶组件，在所述高阶组件中执行以下步骤：获取数据，所述数据和页面元素相关；计算所述数据的嵌套层级；比较所述嵌套层级和预设阈值；根据比较结果选择比对方式，进行数据差异比对；以及根据比对结果渲染所述组件。该方法针对获取到的数据进行智能判断，并针对不同的数据采用不同的差异比对方法，以此提高差异比对的效率，提高系统整体性能。



CN 105808237 A

1.一种页面渲染方法，包括创建高阶组件，在所述高阶组件中执行以下步骤：

获取数据，所述数据和页面元素相关；

计算所述数据的嵌套层级；

比较所述嵌套层级和预设阈值；

根据比较结果选择比对方式，进行数据差异比对；以及

根据比对结果渲染所述组件。

2.根据权利要求1所述的页面渲染方法，其中，所述根据比较结果选择比对方式进行差异数据差异比对包括：

如果所述嵌套层级不大于预设阈值，通过第一插件进行数据差异比对；

如果所述嵌套层级大于预设阈值，通过第二插件进行数据差异比对，

其中，所述第一插件和第二插件分别在所述嵌套层级中具有优化的执行效率。

3.根据权利要求1所述的页面渲染方法，其中，所述根据比对结果渲染所述组件包括：

如果所述比对结果为所述数据发生变化，则执行所述组件的渲染方法；以及

如果所述比对结果为所述数据未变化，则不执行所述组件的渲染方法。

4.根据权利要求1所述的页面渲染方法，还包括：根据多个所述高阶组件更新渲染树。

5.一种页面渲染系统，包括：多个高阶组件容器，所述高阶组件容器中包含数据获取单元、组件判断单元和渲染单元，所述数据获取单元用于获取用户数据，所述组件判断单元通过计算所述数据的嵌套层级，比较所述嵌套层级和预设阈值，根据比较结果进行数据差异化比对，所述渲染单元根据比对结果渲染高阶组件。

6.根据权利要求5所述的页面渲染方法，其中，所述组件判断单元包括：

如果所述嵌套层级不大于预设阈值，通过第一插件进行数据差异比对；

如果所述嵌套层级大于预设阈值，通过第二插件进行数据差异比对，

其中，所述第一插件和第二插件分别在所述嵌套层级中具有优化的执行效率。

7.根据权利要求5所述的页面渲染方法，其中，所述渲染单元包括：

如果所述比对结果为所述数据发生变化，则执行所述组件的渲染方法；以及

如果所述比对结果为所述数据未变化，则不执行所述组件的渲染方法。

8.根据权利要求5所述的页面渲染方法，还包括：DOM更新单元，用于根据多个所述高阶组件更新渲染树。

页面渲染方法和页面渲染系统

技术领域

[0001] 本发明涉及计算机技术领域，具体涉及一种页面渲染方法和页面渲染系统。

背景技术

[0002] 在传统的WEB项目中，前端渲染技术多数采用传统的渲染机制，而这种机制带来的弊端是每当用户与网页交互或者通过脚本程序修改网页时，渲染树包括(DOM文档对象)渲染模型和(CSS层叠样式)渲染模型都会进行Repaint(重绘)或者Reflow(重排)，产生这种现象是因为网页的内部结构已经发生了改变，这种改变带给浏览器的是性能损耗，资源占用，带给用户的是响应时长的延长。

[0003] 基于此，Facebook推出的用于构建用户界面的React工具集。通过React工具集，系统能够构建和网页元素对应的组件，根据用户在网页元素上的操作或网页元素的属性变化更新组件，然后根据组件更新渲染树，以完成整个真实页面的渲染。React脚本库和Flux单项数据流架构结合能有效地提高渲染效率，降低浏览器的响应时间。

[0004] 但React渲染技术中有其缺点。由于React技术采用脏检测机制，如果虚拟组件里的某个节点数据发生变化时，如果不能确定节点数据所造成的影响范围，则默认该节点和其所有的子节点都发生了变化。如图1所示，节点B的数据发生变化，则默认子节点D、E、H、I、K都可能发生变化，当更新节点B的时候，默认更新节点D、E、H、I、K，由此带来许多不必要的渲染(即，更新没有变化的数据)，导致渲染效率的降低。

发明内容

[0005] 有鉴于此，本发明提供一种页面渲染方法和页面渲染系统，提高页面的渲染效率。

[0006] 根据本发明的第一方面，提供一种页面渲染方法，包括创建高阶组件，在所述高阶组件中执行以下步骤：获取数据，所述数据和页面元素相关；计算所述数据的嵌套层级；比较所述嵌套层级和预设阈值；根据比较结果选择比对方式，进行数据差异比对；以及根据比对结果渲染所述组件。

[0007] 优选地，所述根据比较结果选择比对方式进行差异数据差异比对包括：如果所述嵌套层级不大于预设阈值，通过第一插件进行数据差异比对；如果所述嵌套层级大于预设阈值，通过第二插件进行数据差异比对，其中，所述第一插件和第二插件分别在所述嵌套层级中具有优化的执行效率。

[0008] 优选地，所述根据比对结果渲染所述组件包括：如果所述比对结果为所述数据发生变化，则执行所述组件的渲染方法；以及如果所述比对结果为所述数据未变化，则不执行所述组件的渲染方法。

[0009] 优选地，还包括：根据多个所述高阶组件更新渲染树。

[0010] 根据本发明的第二方面，提供一种页面渲染系统，包括：多个高阶组件容器，所述高阶组件容器中包含数据获取单元、组件判断单元和渲染单元，所述数据获取单元用于获取用户数据，所述组件判断单元通过计算所述数据的嵌套层级，比较所述嵌套层级和预设

阈值，根据比较结果进行数据差异化比对，所述渲染单元根据比对结果渲染高阶组件。

[0011] 优选地，所述组件判断单元包括：如果所述嵌套层级不大于预设阈值，通过第一插件进行数据差异比对；如果所述嵌套层级大于预设阈值，通过第二插件进行数据差异比对，其中，所述第一插件和第二插件分别在所述嵌套层级中具有优化的执行效率。

[0012] 优选地，所述渲染单元包括：如果所述比对结果为所述数据发生变化，则执行所述组件的渲染方法；以及如果所述比对结果为所述数据未变化，则不执行所述组件的渲染方法。

[0013] 优选地，还包括：DOM更新单元，用于根据多个所述高阶组件更新渲染树。

[0014] 本实施例提供的页面渲染方法，包括创建高阶组件，在所述高阶组件中执行以下步骤：获取数据，所述数据和页面元素相关；计算所述数据的嵌套层级；比较所述嵌套层级和预设阈值；根据比较结果选择比对方式，进行数据差异比对；以及根据比对结果渲染所述组件。该方法针对获取到的数据进行智能判断，并针对不同的数据采用不同的差异比对方法，以此提高差异比对的效率，提高系统整体性能。

附图说明

[0015] 通过参照以下附图对本发明实施例的描述，本发明的上述以及其它目的、特征和优点将更为清楚，在附图中：

[0016] 图1是现有技术中的组件的节点数据的示意图；

[0017] 图2是现有技术中的Flux单项数据流架构的示意图；

[0018] 图3是根据本发明实施例的页面渲染方法的流程图；

[0019] 图4是根据本发明实施例的页面渲染系统的结构图。

具体实施方式

[0020] 以下基于实施例对本发明进行描述，但是本发明并不仅仅限于这些实施例。在下文对本发明的细节描述中，详尽描述了一些特定的细节部分。对本领域技术人员来说没有这些细节部分的描述也可以完全理解本发明。为了避免混淆本发明的实质，公知的方法、过程、流程没有详细叙述。另外附图不一定是按比例绘制的。

[0021] 附图中的流程图、框图图示了本发明实施例的系统、方法、装置的可能的体系框架、功能和操作，流程图和框图上的方框可以代表一个模块、程序段或仅仅是一段代码，所述模块、程序段和代码都是用来实现规定逻辑功能的可执行指令。也应当注意，所述实现规定逻辑功能的可执行指令可以重新组合，从而生成新的模块和程序段。因此附图的方框以及方框顺序只是用来更好的图示实施例的过程和步骤，而不应以此作为对发明本身的限制。

[0022] 系统的各个模块或单元可以通过硬件、固件或软件实现。软件例如包括采用JAVA、C/C++/C#、SQL等各种编程语言形成的编码程序。虽然在方法以及方法图例中给出本发明实施例的步骤以及步骤的顺序，但是所述步骤实现规定的逻辑功能的可执行指令可以重新组合，从而生成新的步骤。所述步骤的顺序也不应该仅仅局限于所述方法以及方法图例中的步骤顺序，可以根据功能的需要随时进行调整。例如将其中的某些步骤并行或按照相反顺序执行。

[0023] 图2是现有技术中的Flux单项数据流架构的示意图。用户在页面的操作或页面脚本触发一个动作(Action),该动作触发一个事件(Trigger event)并将事件传递到分发器(Dispatcher),分发器(Dispatcher)分发和装载视图(Dispatch payload)到数据仓库(Store),数据仓库(Store)用于存储数据、维护数据操作,数据仓库(Store)向分发器返回注册信息(register callback),数据仓库将接收到的数据改动传播(emit change)到视图(View),视图处于侦听状态,一旦侦听到数据,从数据仓库中获取数据,并根据数据的改动情况完成页面渲染。Flux架构的单项数据流在大规模的WEB应用中要比采用MVC架构的双向绑定机制性能更高,易于维护与理解,耦合度大幅度降低。

[0024] 图3是根据本发明实施例的页面渲染方法的流程图。所述方法包括步骤300-步骤306。

[0025] 在步骤300中,创建高阶组件。组件是React工具库内引入的概念,根据页面元素创建对应的组件,在用户修改或删除页面元素时,修改或删除和页面元素对应的组件。高阶组件(Higher order component)和上述的组件相对,是上述组件的扩展类,用于封装一个底层的组件类,返回新的高阶组件类。以下的301-306步骤都在高阶组件类中执行。

[0026] 在步骤301中,获取数据。该数据为和页面元素相关的数据。

[0027] 当使用Flux架构时,该数据存储在数据仓库中。Flux架构能将用户在页面的操作或页面脚本触发的动作传递到分发器,分发器将相关数据分发到数据仓库,数据仓库对数据处理后,将数据通知到对应的虚拟组件。

[0028] 在本步骤中,组件获取到页面元素相关的数据。例如,文本框组件获取到用户通过该文本框输入的文本内容,或者表格组件获取当前页面需要展示的数据。组件获得的数据格式多种多样,可以是基础格式,也可以是复杂格式。基础格式如字符串,整型或浮点型数值等,一级数组,复杂数据为多种基础格式数据的组合,例如,下面的两级嵌套的JASON格式数据。

[0029] {

[0030] "people":[

[0031] {"firstName":"Brett","lastName":"McLaughlin","email":"aaaa"},

[0032] {"firstName":"Jason","lastName":"Hunter","email":"bbbb"},

[0033] {"firstName":"Elliotte","lastName":"Harold","email":"cccc"}

[0034]]

[0035] }

[0036] 在步骤302中,计算数据的嵌套层级。

[0037] 在本步骤中,计算在步骤301中获取到的数据嵌套层级。例如,上面的JASON格式数据为两级嵌套数据。

[0038] 在步骤303中,判断嵌套层级是否超过预设阈值。如果嵌套层级不超过预设阈值,则执行步骤304,否则执行步骤305。

[0039] 在步骤304中,通过第一插件进行数据差异比对。在本步骤中,使用插件,例如PureRenderMixin,对嵌套层级不超过预设阈值的数据进行差异性比对,以确定本次数据是否发生变化。

[0040] 插件PureRenderMixin为Facebook推出的插件。在该插件的方法

shouldComponentUpdate中，实现了当前接收到的数据和上一次数据的差异性比较，当两者相等的时候返回值为false，表示数据没有发生变化，否则返回值为true，表示本次接收到的数据和上一次的数据发生了变化。但遗憾的是，PureRenderMixin并不能很好的进行对象的比较。它只会检查对象引用的相等性(===)，也就是说，对于有相同数据的不同对象而言它会返回false。例如，如果shouldComponentUpdate包含下面的代码，

[0041] var a={foo:'bar'};

[0042] var b={foo:'bar'};

[0043] a===b;//false

[0044] 可以看到，变量a和b的数据是相同的，但它们隶属于不同对象的引用，因此shouldComponentUpdate返回值是false。所以为了比较不同的对象，可以包括类似以下的代码：

[0045] var a={foo:'bar'};

[0046] var b=a;

[0047] b.foo='bar';

[0048] a===b;//true

[0049] 因此在数据嵌套层级较少的情况下，添加类似上述的代码能够完成数据的差异性比对，执行的效率也较高。

[0050] 在步骤305中，通过第二插件进行数据差异比对。在本步骤中，使用插件，例如Immutable，对嵌套层级超过预设阈值的数据进行差异性比对，以确定本次数据是否发生变化。

[0051] 当数据层级较多，嵌套比较复杂时，通过PureRenderMixin进行对象比较就很复杂，不仅需要添加大量的代码(代码增加导致程序错误的概率增加)，执行效率也不是很高。因此对于复杂对象的数据的比较，需要借助Facebook推出的Immutable插件来完成。

[0052] Immutable创建的数据集为不可变数据集，一旦创建就不能被修改，这使得应用开发更为简单，并且使得变化检查逻辑变得简单。例如，

[0053] var foo=immutable.fromJS({a:{b,1}});

[0054] var bar=foo.setIn([a,b],2);

[0055] console.log(foo.getIn([a,b]));//输出1

[0056] console.log(bar.getIn([a,b]));//输出2

[0057] foo===bar;//返回值为false

[0058] 如上，使用Immutable创建的对象可以直接使用===或ImmutableIs来比较，这意味着能够方便快速的进行对象比较。因此，在本步骤中，对于嵌套层级超过阈值的数据通过Immutable插件提供的函数进行本次数据和上一次的数据进行比对，以确定本次数据是否发生变化。

[0059] 在步骤306中，根据比对结果渲染组件。

[0060] 在本步骤中，根据步骤303或304的比对结果，更新组件。即，如果比对结果为true表示数据发生变化，更新组件，如果数据没有发生变化，不需要渲染组件。

[0061] 优选地，根据比对结果渲染所述组件包括：当数据发生变化，则执行组件的渲染方法；以及如果数据未变化时，则不执行组件的渲染方法。

[0062] 在另一个优选的实施例中，所述渲染方法还包括：根据组件更新所述渲染树。在上述的渲染方法中，高阶组件会根据数据变化情况确定是否执行渲染方法，只在数据发生变化时，执行渲染方法。由于每个高阶组件会对应一个页面元素，最终根据高阶组件对渲染树进行更新，从而实现整个网页页面的渲染。

[0063] 在本实施例中，通过创建高阶组件，在高阶组件中执行以下步骤：获取数据，所述数据和页面元素相关；计算数据的嵌套层级；比较嵌套层级和预设阈值；根据比较结果选择比对方式，进行数据差异比对；以及根据比对结果渲染组件。进一步，所述根据比较结果选择比对方式包括：如果嵌套层级不大于预设阈值，如果嵌套层级不大于预设阈值，通过第一插件(例如，PureRenderMixin)进行数据差异比对，如果嵌套层级大于预设阈值，通过第二插件(例如，Immutable)进行数据差异比对，其中，第一插件和第二插件分别在嵌套层级中具有优化的执行效率。通过高阶组件封装插件Immutable和插件PureRenderMixin，在获得优化的比对效率的同时，解决ES6语法不支持Mixin组件的问题。ES6是JavaScript语言的版本。它对JavaScript做了大量改造，提高了灵活性和应用性，使得这门语言真正成为了企业级开发工具。

[0064] 图4是根据本发明实施例的页面渲染系统的结构图。该页面渲染系统包括：多个高阶组件容器401和DOM更新单元402。高阶组件容器401中和页面元素对应，分别包括数据获取单元4010、组件判断单元4011和渲染单元4012。

[0065] 数据获取单元4010用于获取用户数据，组件判断单元4011通过计算数据的嵌套层级，比较所述嵌套层级和预设阈值，根据比较结果选择比较方式(第一插件或第二插件)，进行数据差异化比对；渲染单元4012根据比对结果渲染虚拟组件。DOM更新单元402根据多个高阶组件容器401更新渲染树，DOM更新单元402通过React提供的接口完成。

[0066] 在一个优选的实施例中，上述组件判断单元包括：如果嵌套层级不大于预设阈值，通过第一插件进行数据差异比对；如果嵌套层级大于预设阈值，通过第二插件进行数据差异比对，其中，所述第一插件和第二插件分别在所述嵌套层级中具有优化的执行效率。

[0067] 在一个优选的实施例中，组件判断单元为高阶组件容器的组件更新方法(shouldComponentUpdate)，所述渲染单元为高阶组件容器的渲染(render)方法。

[0068] 在一个优选的实施例中，所述渲染单元包括：如果所述比对结果为所述数据发生变化，则执行所述组件的渲染方法；以及如果所述比对结果为所述数据未变化，则不执行所述组件的渲染方法。

[0069] 本实施例提供的页面渲染系统，包括：高阶组件容器和DOM更新单元，高阶组件容器中包含组件判断单元和渲染单元，所述组件判断单元通过计算数据的嵌套层级，比较所述嵌套层级和预设阈值，根据比较结果进行数据差异化比对；所述渲染单元根据比对结果渲染虚拟组件；DOM更新单元用于根据虚拟组件更新DOM文档对象。该页面渲染系统对高阶组件进行重构，在组件里判断嵌套层级，从而确定差异比较方法，实现优化的差异比较，不需要添加大量的代码，减少程序出错的概率，提高页面渲染的性能。

[0070] 根据本发明的系统和方法可以部署在单个或多个服务器上。例如，可以将不同的模块分别部署在不同的服务器上，形成专用服务器。或者，可以在多个服务器上分布式部署相同的功能单元、模块或系统，以减轻负载压力。所述服务器包括但不限于在同一个局域网以及通过Internet连接的多个PC机、PC服务器、刀片机、超级计算机等。

[0071] 以上所述仅为本发明的优选实施例，并不用于限制本发明，对于本领域技术人员而言，本发明可以有各种改动和变化。凡在本发明的精神和原理之内所作的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。

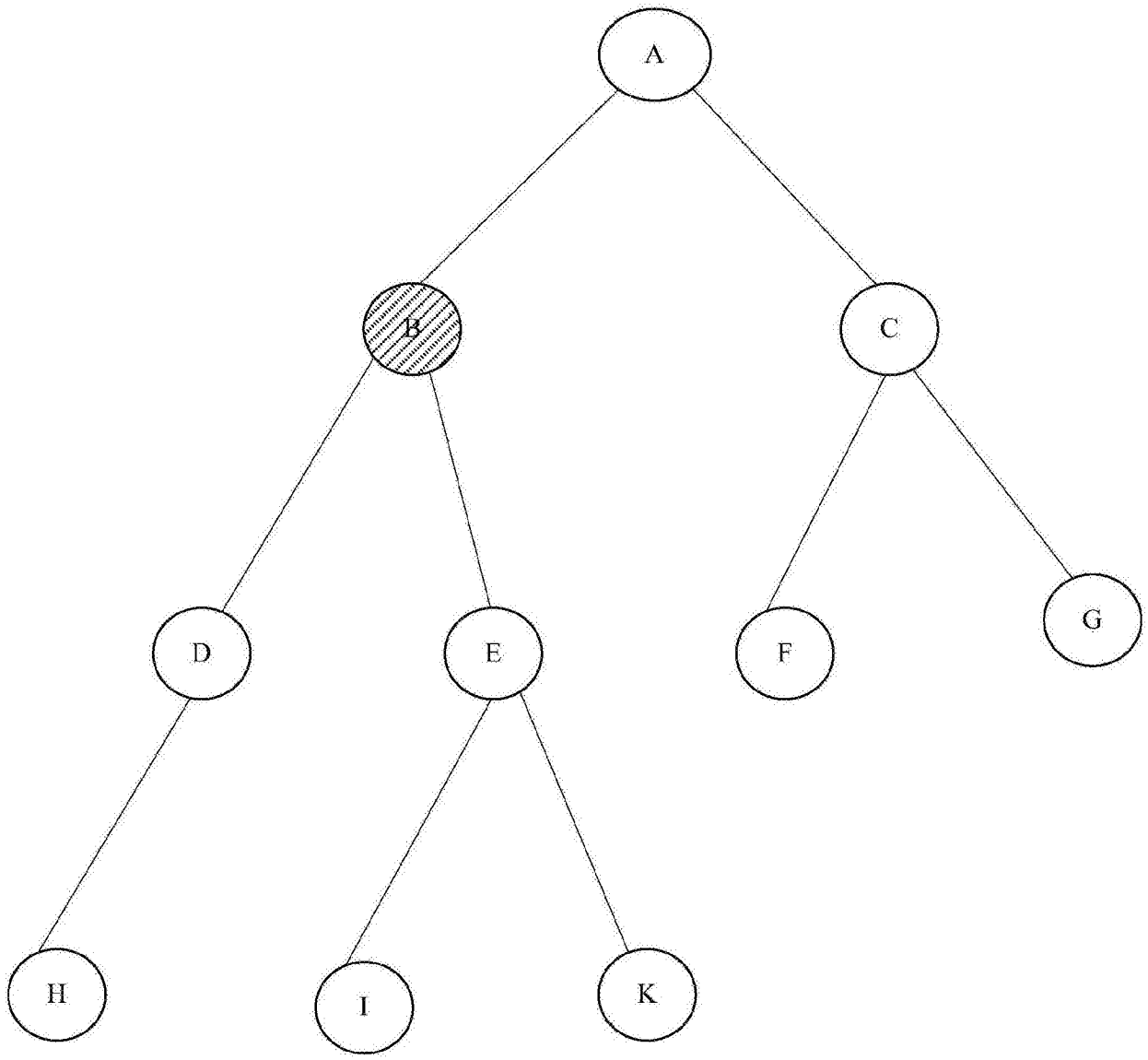


图1

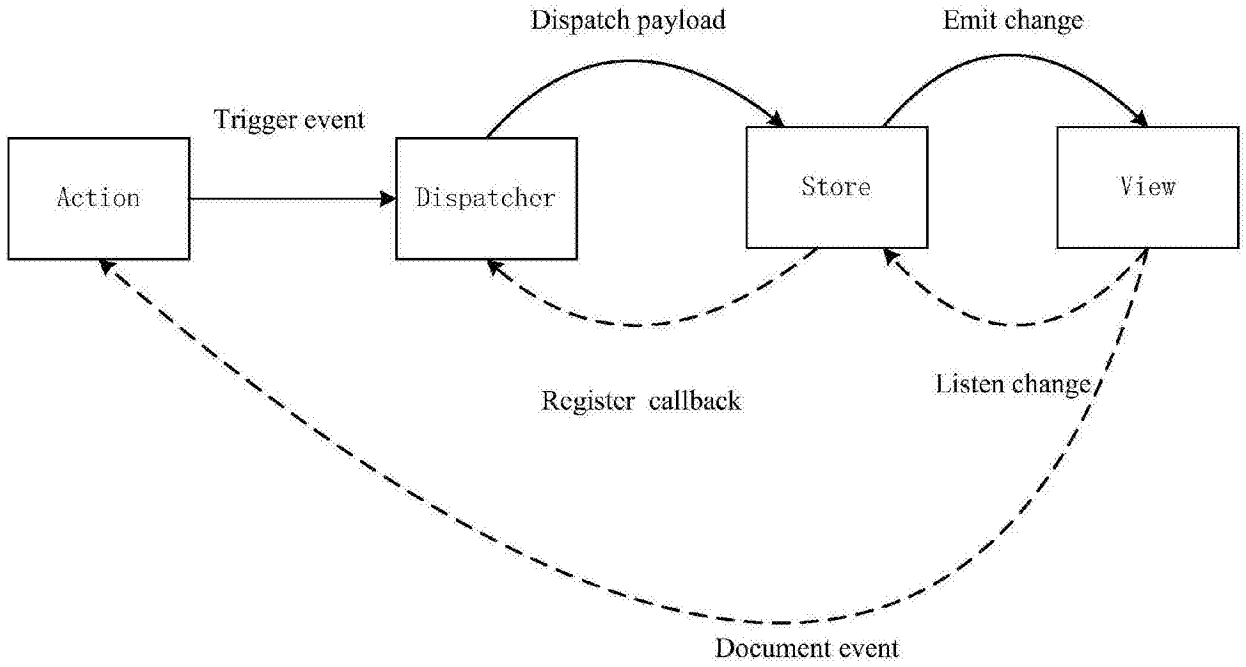

Dispatch payload
Emit change
Trigger event
Action
Dispatcher
Store
View
Register callback
Listen change
Document event

图2

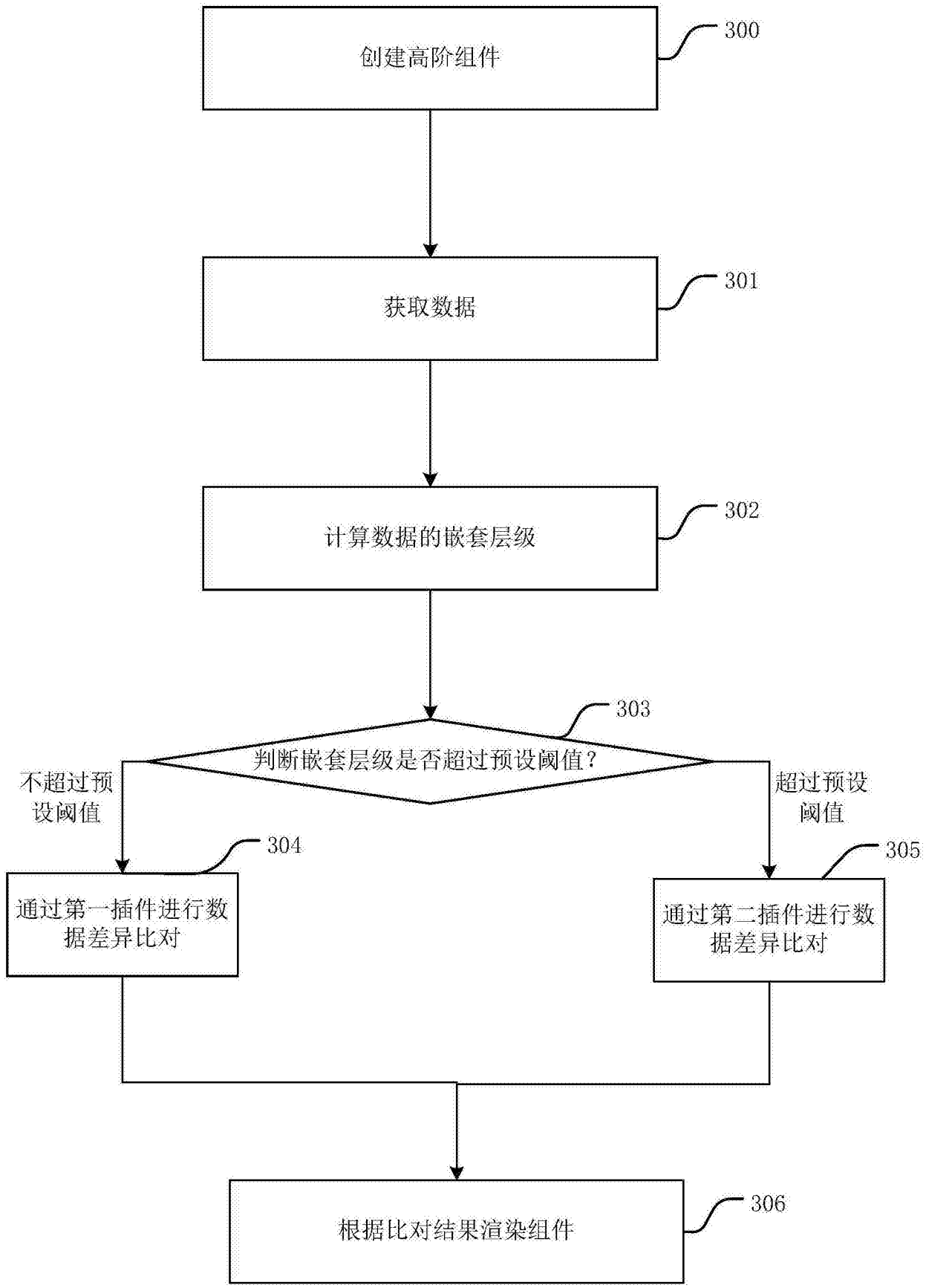

创建高阶组件
300
获取数据
301
计算数据的嵌套层级
302
判断嵌套层级是否超过预设阈值？
303
不超过预设阈值
超过预设阈值
通过第一插件进行数据差异比对
304
通过第二插件进行数据差异比对
305
根据比对结果渲染组件
306

图3

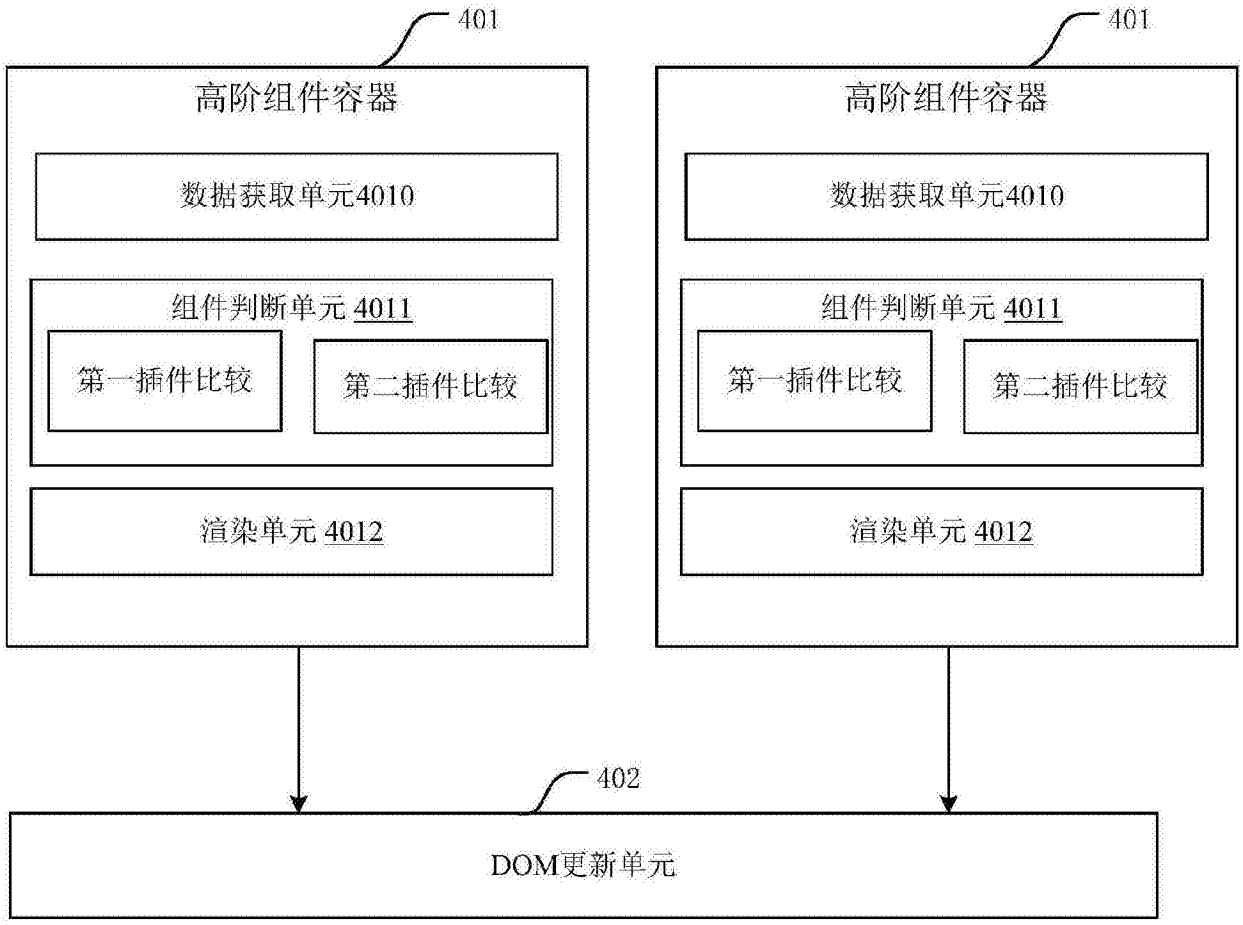

401
高阶组件容器
数据获取单元4010
组件判断单元 4011
第一插件比较
第二插件比较
渲染单元 4012
401
高阶组件容器
数据获取单元4010
组件判断单元 4011
第一插件比较
第二插件比较
渲染单元 4012
402
DOM更新单元

图4