Vue+Echarts 监控大屏实例：软件系统运行监控模板实例

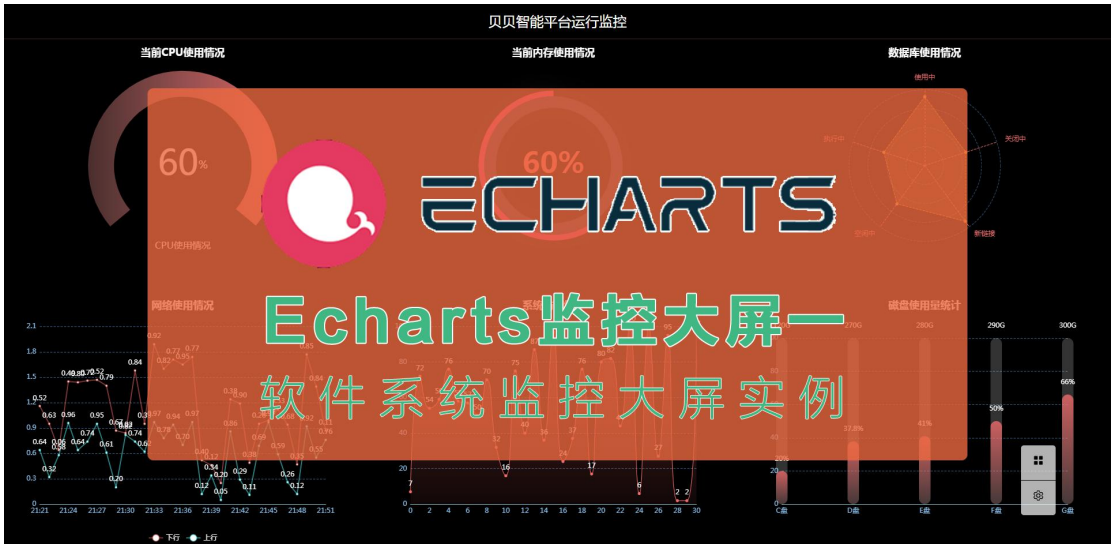
一、实例概述

本实例实现软件系统运行状况监控大屏实例，包括数据库状态监控、系统运行内存、网络、cpu 及磁盘监控等各种系统运行状况监控界面，实现系统运行状况监控可视化。本实例实现对于监控界面的相关开发资料，提供实例源码、开发过程视频及实现过程。

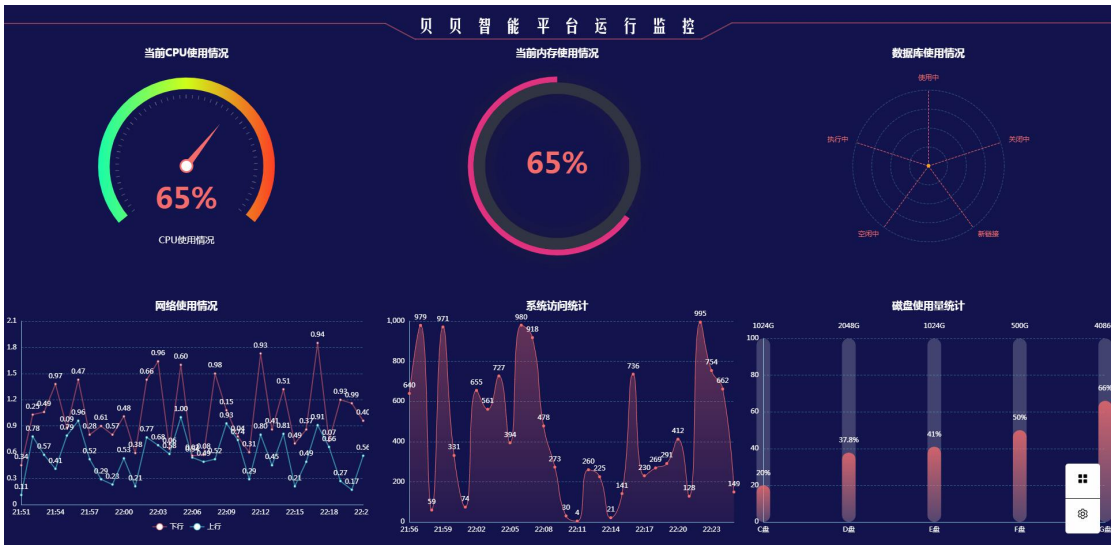
界面使用 vue+echarts 开发，数据使用 json 格式文件进行模拟，web 服务器使用 nginx 进行搭建。

1. **Vue 参考文档：**https://v3.cn.vuejs.org/
2. **Echarts 参考文档：**https://echarts.apache.org/zh/index.html
3. **Element-UI 参考文档：**https://element-plus.org/zh-CN/#/zh-CN
4. **开发工具：**HBuilder X3.3.11.20220209
5. **Web 服务器：**Nginx

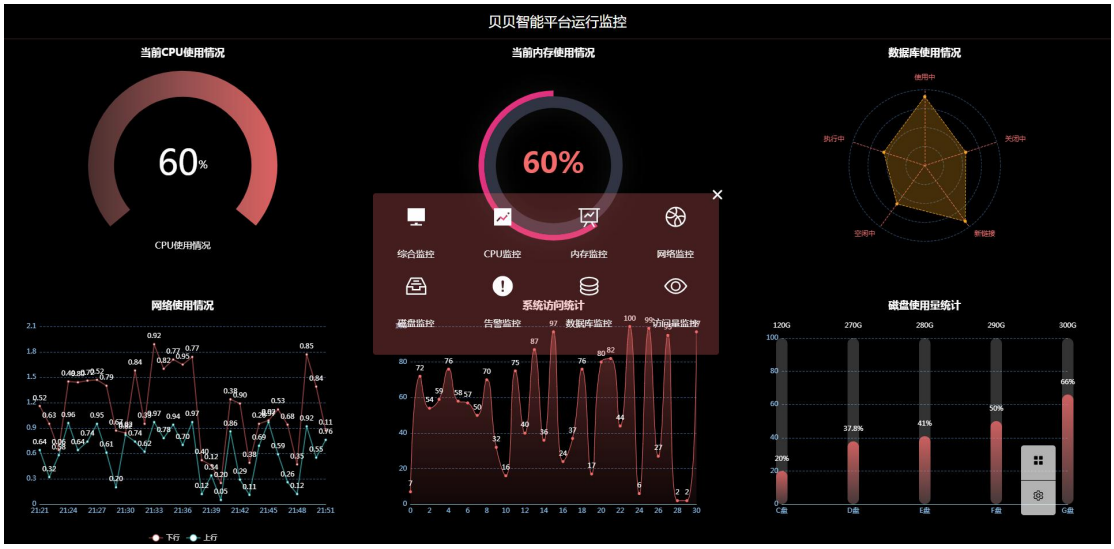
二、效果预览



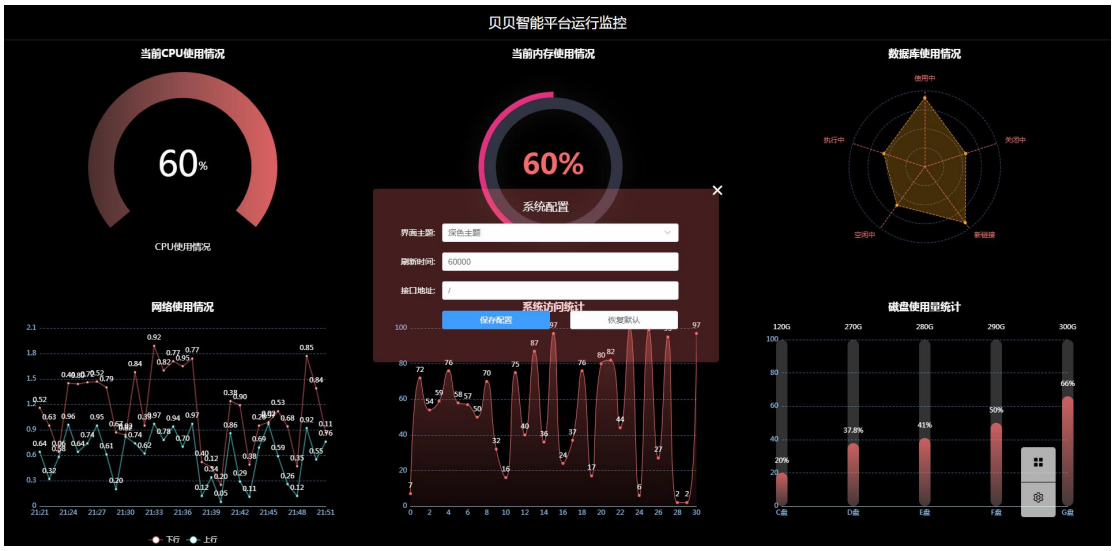
1. 大屏首页



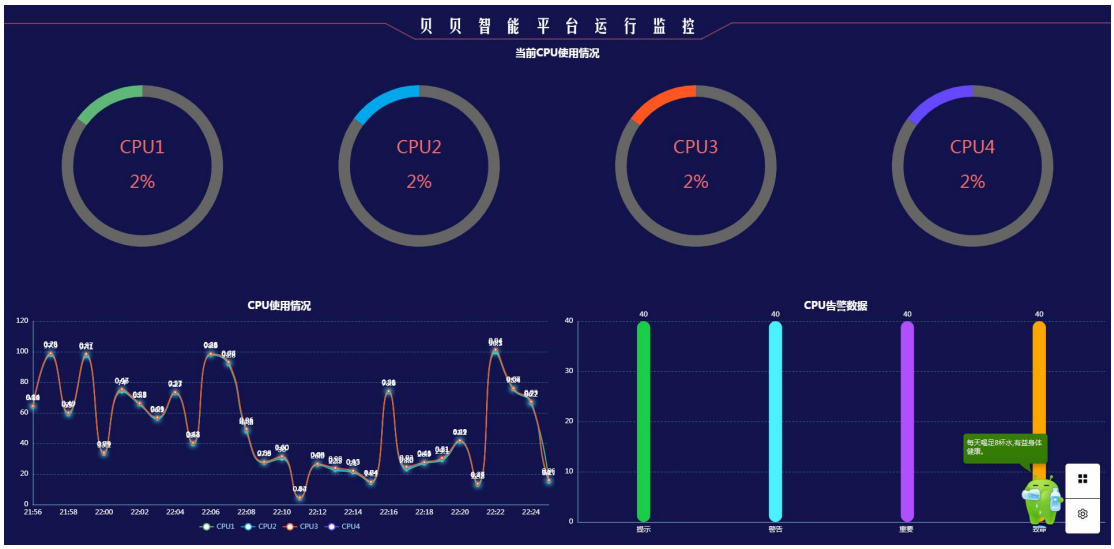
2. 菜单界面



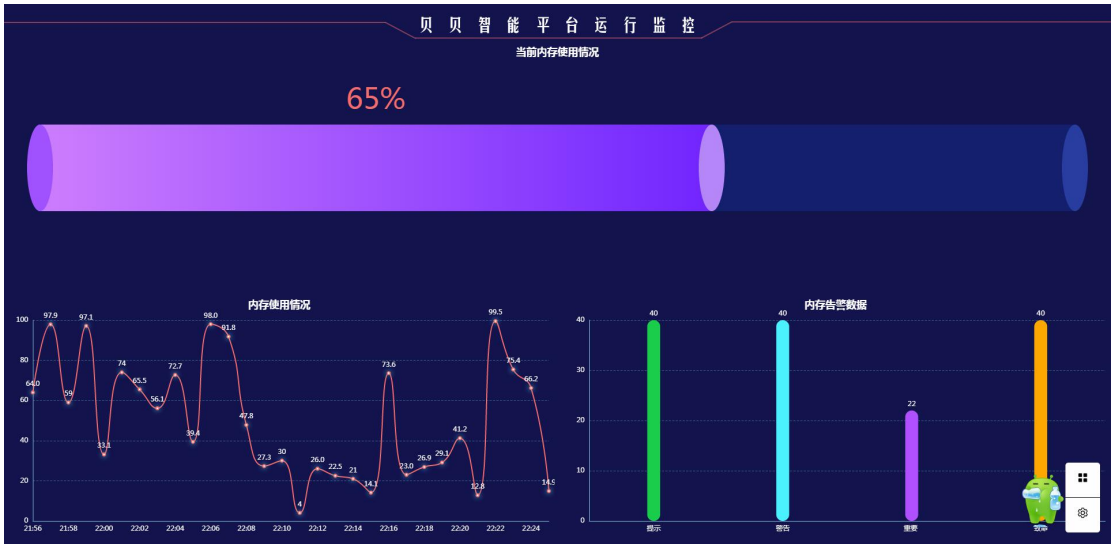
3. 配置界面



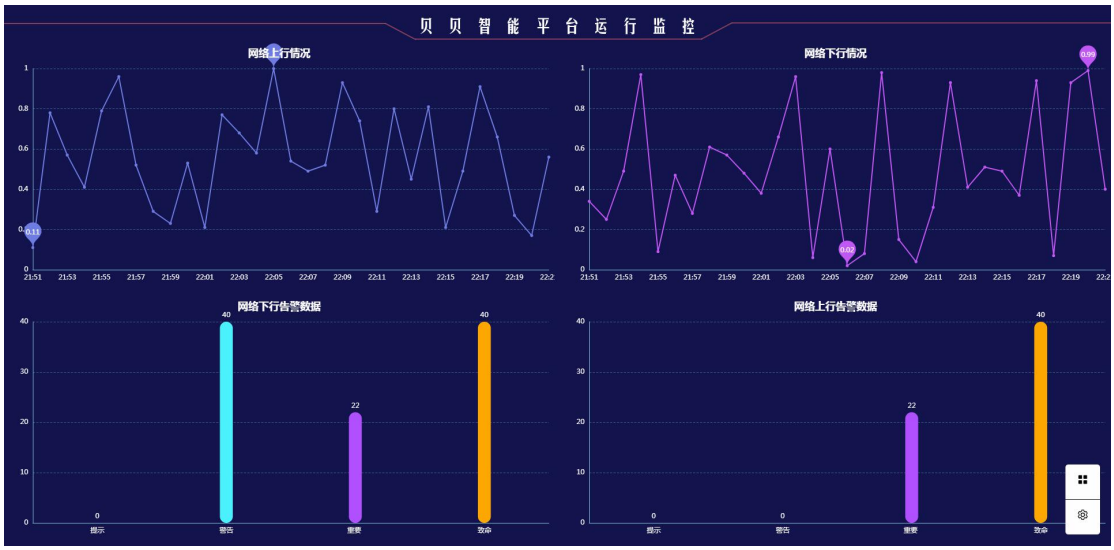
4. CPU 界面



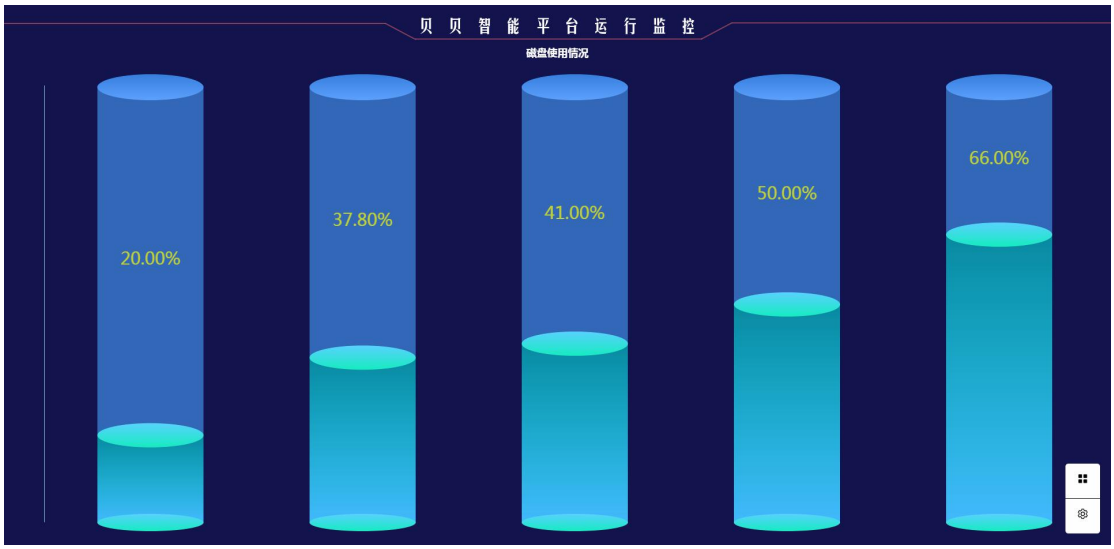
5. 内存界面



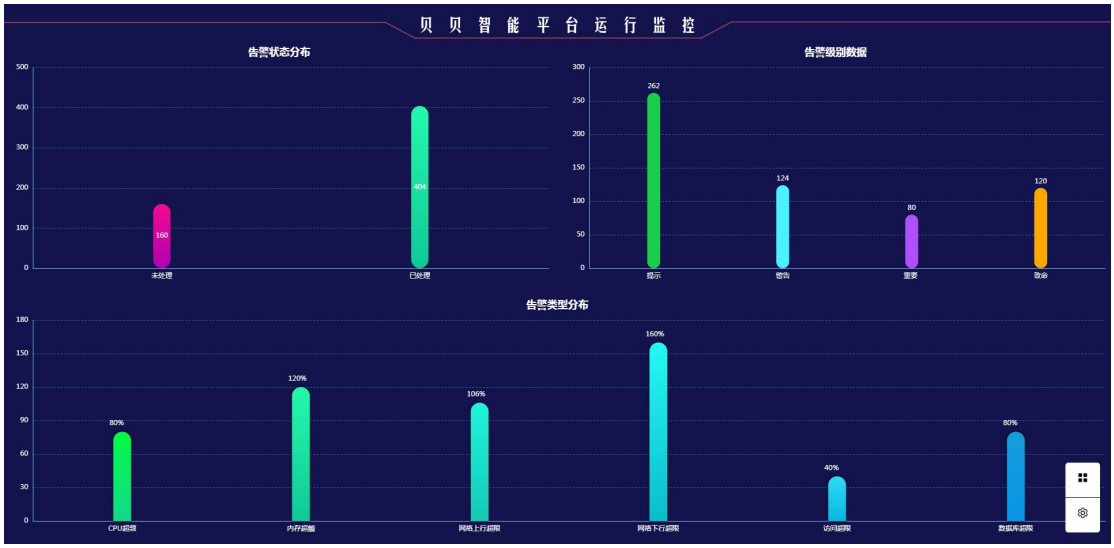
6. 网络界面



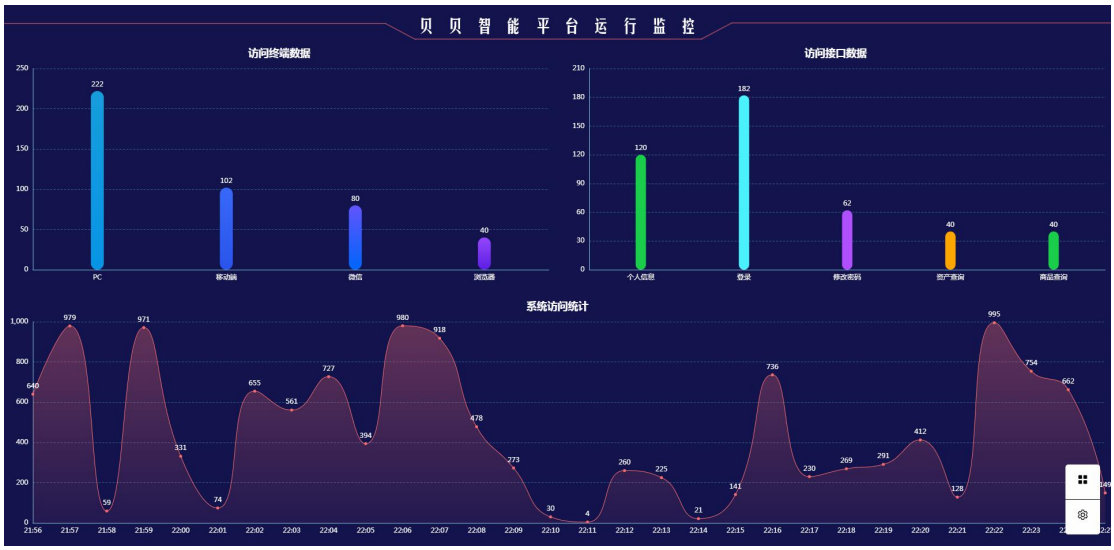
7. 磁盘界面



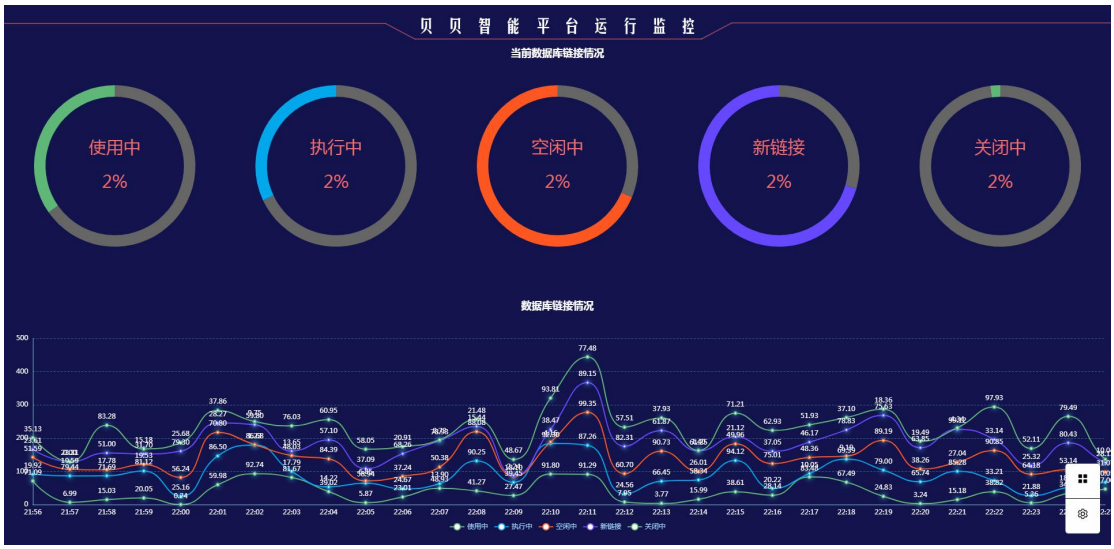
8. 告警界面



9. 访问量界面



10. 数据库界面



三、开发视频

VUE+Echarts 监控大屏实例一：软件系统运行监控大屏模版

四、使用场景

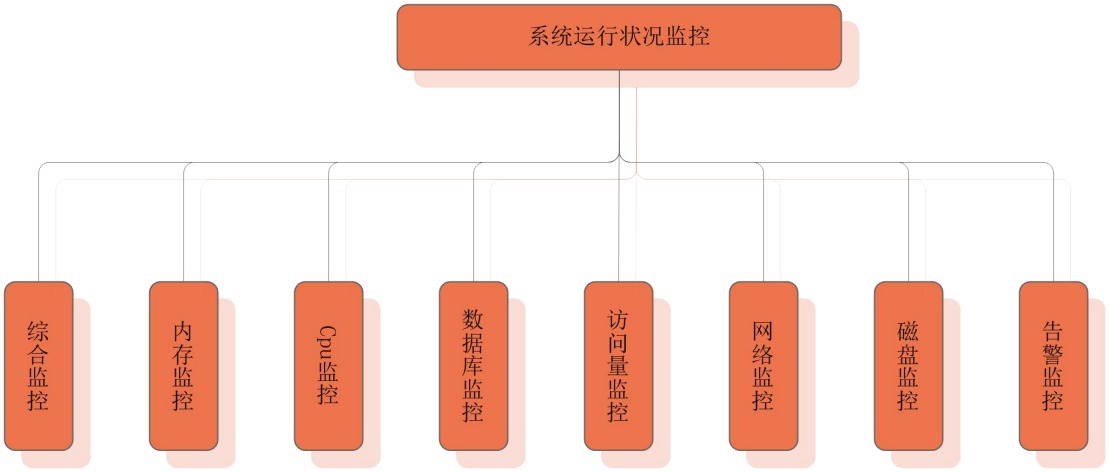
系统运维过程中使用，可通过这些监控项目检测系统运行状况，系统运行健康状况，查看系统运行压力、系统告警信息等。

可通过此实例学习相关前端开发知识，也可以将此实例当做模板参考在实际项目中进行使用。

五、界面规划

实现综合监控、数据库监控、磁盘监控、Cpu 监控、网络监控、内存监控、访问量监控、告警监控等监控界面。

界面具体的分辨率需要根据实际大屏的分辨率进行设定，若使用此实例中的模板，请根据实际情况进行调整。



另外除了展示界面外提供展示配置界面，可以配置界面主题、字体、布局、动态数据刷新时间等。

六、主题规划

界面主题可通过配置选项进行自动切换，规划后展示不同主题对应的效果，可以根据实际需求进行调整和定制。

6.1 深色主题

内容	值	备注
背景色	#13134e	页面背景颜色
字体颜色	#ffffff	页面字体颜色，主要使用白色字体
图表颜色	#85C1D9	页面监控图表相关颜色（线条、轴等），根据实际情况设置。
其他	--	根据实际情况搭配

6.2 深蓝主题

内容	值	备注
背景色	#0f197f	页面背景颜色
字体颜色	#ffffff	页面字体颜色，主要使用白色字体
图表颜色	#85C1D9	页面监控图表相关颜色（线条、轴等），根据实

		际情况设置。
其他	--	根据实际情况搭配

七、界面明细

7.1 展示配置界面

本界面可通过可视化的方式配置展示时的一些属性，包括主题、动态数据加载周期等内容，可通过本配置实现监控界面的展示效果自定义，通过悬浮按钮及弹框的方式进行展示，数据存储在浏览器 store 中，每次加载自动调用。

7.2 综合监控界面

综合监控界面展示系统名称、当前访问量、当前内存、当前 cpu、当前磁盘、当前数据库链接数等。

序号	图表	类型
1	当前访问量	折线图，展示一段时间内系统的访问量走势
2	内存使用量	饼图，展示当前系统内存使用情况
3	cpu 使用量	仪表盘，展示当前系统 cpu 使用情况
4	磁盘使用量	柱状图，展示当前系统磁盘使用情况
5	数据库链接状况	雷达图，展示当前系统数据库链接状况

7.3 内存监控界面

系统服务器内存监控，展示内存当前使用情况、历史使用情况、超预警时长统计等内容。

序号	图表	类型
1	当前使用量	柱状图
2	内存使用记录	折线图，30 分钟内内存使用情况统计图
3	内存告警情况	柱状图，告警级别统计

7.4 CPU 监控界面

系统服务器 CPU 监控界面，展示当前系统服务器 CPU 使用情况，历史使用情况、超预警时长统计等。

序号	图表	类型
1	当前使用量	多饼图，展示各 cpu 的使用情况
2	使用记录	折线图，30 分钟内各 cpu 使用记录图
3	告警情况	柱状图，告警级别统计

7.5 数据库监控界面

当前系统数据库链接情况，展示当前打开的连接数、真正使用的链接数等数据库监控报表。

序号	图表	类型
1	当前使用量	多饼图，各种类型数据库链接的分布情况
2	使用记录	折线图
3	当前使用量	多饼图，各种类型数据库链接的分布情况

7.6 访问量监控界面

系统访问量监控展示，包括访问次数、访问接口排名、访问人数等各项指标的监控情况。

序号	图表	类型
1	访问终端统计	柱状图，各接口访问情况统计
2	访问接口统计	柱状图，各接口访问情况统计
3	访问量统计	折线图，30 分钟内访问总量统计结果
5	访问终端统计	柱状图，各接口访问情况统计

7.7 网络监控界面

系统网络情况监控展示，包括上行网络、下行网络、峰值、超预警时长统计等各项指标展示。

序号	图表	类型
1	上行情况	折线图，请求网络流量统计情况
2	下行情况	折线图，下发数据流量统计情况
3	上行告警	柱状图，超预警报警数据统计
4	下行告警	柱状图，超预警报警数据统计

7.8 磁盘监控界面

系统运行服务器磁盘状况监控，包括磁盘数量、磁盘容量、使用量等各项指标的报表展示。

序号	图表	类型
1	磁盘当期状况	柱状图，展示当前磁盘使用量以及总量

7.9 告警监控界面

系统告警监控界面，包括系统 cpu 超预警统计报表、内存超预警统计报表、磁盘超预警统计报表、数据库链接超预警统计报表、网络超预警统计报表及异常事件统计报表等。

序号	图表	类型
1	告警状态分布	柱状图，不同状态告警数据量统计
2	告警类型分布	柱状图，不同类型告警数据量统计
3	告警级别分布	柱状图，不同级别告警数据量统计

八、代码实现

本实例使用 HBuilder X 开发工具开发，使用 vue+echarts 实现对应图表。

8.1 框架搭建

使用 HBuilderX 创建 vue3 项目，并创建公共文件夹、资源文件夹、页面文件夹等，引入必要的依赖库，如下：

1. **Jquery：**

用于界面元素操作，版本：3.6.0。

安装指令：

npm install jquery --save

代码引入：

import $ from 'jquery'

2. **Roter：**

用于界面路由控制，版本：4.0.14。

安装指令：

npm install vue-router@4

代码引入：

```
import {
    createRouter,
    createWebHashHistory
} from "vue-router";
const routes = [
  { path: '/', component: Index },
  { path: '/Index', component: Index },
  { path: '/Memory', component: Memory },
  { path: '/Cpu', component: Cpu },
  { path: '/Database', component: Database },
  { path: '/View', component: View },
  { path: '/Net', component: Net },
  { path: '/Dist', component: Dist },
  { path: '/Alert', component: Alert }
]
const router = createRouter({
  // 4. 内部提供了 history 模式的实现。为了简单起见，我们在这里使用 hash 模式。
  history: createWebHashHistory(),
  routes, // `routes: routes` 的缩写
})
app.use(router);
```

3. **Axios：**

用于网络请求，版本：0.26.1。

安装指令：

npm install axios --save

代码引入：

import axios from 'axios';

4. **Echats：**

用于报表展示，版本：5.3.1。

安装指令：

```
npm install echarts -S
```

代码引入：

```
import * as echarts from 'echarts'
```

5. **Element plus：**

用于界面样式控制，版本：2.1.3。

安装指令：

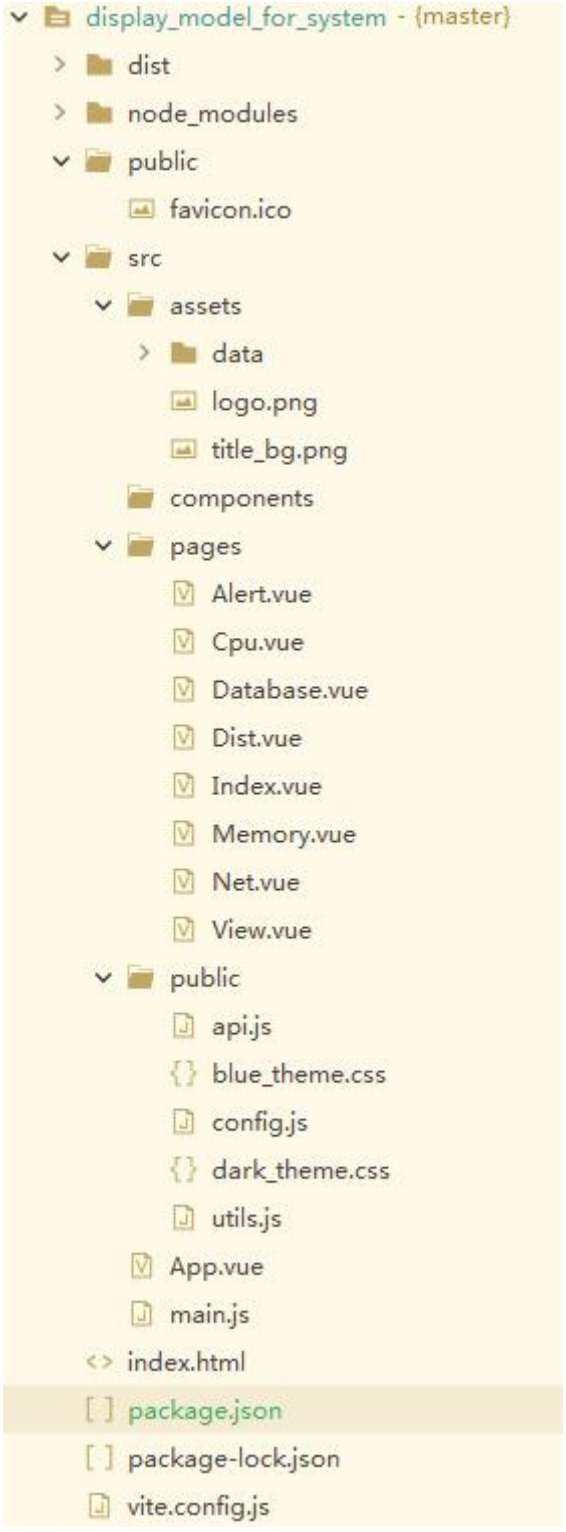
```
npm install element-plus --save
```

代码引入：

```
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'
```

8.2 目录说明

1. 目录结构：



2. 目录说明：

序号	文件	说明
1	./public	公共图标文件夹
2	./src/assets/	资源文件夹
3	./src/assets/data	json 数据文件夹
4	./src/components	VUE 组件目录
5	./src/pages/	界面文件目录
6	./src/pages/Alert.vue	告警监控界面
7	./src/pages/Cpu.vue	CPU 监控界面
8	./src/pages/Database.vue	数据库监控界面
9	./src/pages/Dist.vue	磁盘监控界面
10	./src/pages/Index.vue	综合监控界面
11	./src/pages/Memory.vue	内存监控界面
12	./src/pages/Net.vue	网络监控界面
13	./src/pages/View.vue	访问量监控界面
14	./src/public	公共方法及资源文件夹
15	./src/public/api.js	数据加载接口
16	./src/public/blue_theme.css	蓝色主题样式
17	./src/public/config.js	配置工具方法
18	./src/public/dark_theme.css	深色主题方法
19	./src/public/utils.js	公共方法
20	./src/App.vue	主页
21	./src/main.js	vue 入口
22	./src/index.html 入口文件	

8.3 界面实现

8.3.1 公共方法及样式等整理

1. 主题样式：

此处只展示了两种样式，后续可根据需要进行自定义扩展。

- public/dark_default.css：实现黑色主题相关样式的定义实现。

```
/**
 * 深色主题
 */
@charset "utf-8";

html,body{
    margin: 0;
    padding: 0;
    width: 100%;
    color: #fff;
    overflow: hidden;
}

.dark{
    position: fixed;
    left: 0;
    top:0;
    width: 100%;
    height: 100%;
    background: #13134e;
}

.dark .menu_btn{
    position: fixed;
    right: 1%;
    bottom: 20px;
    width: 60px;
    color: #000000;
}
.dark .menu_btn_item{
    height: 60px;
    line-height: 60px;
```

```
        text-align: center;
        background: rgba(255,255,255,1);
        cursor: pointer;
}

.dark .menu_btn_item:hover{
        background: #F56C6C;
        color:#fff;
}
.dark .title{
        text-align: center;
        color: #fff;
        font-size: 24px;
        /* border-bottom: 1px solid #f56c6c47; */
}

.dark .menu_panel{
}

.dark .menu_panel .menu_panel_maker{
        position: fixed;
        left: 0px;
        top: 0px;
        width: 100%;
        height: 100%;
        z-index: 1;
}

.dark .menu_panel .inner_contaner{
        position: fixed;
        background: #f56c6c47;
        border-radius: 5px;
        z-index: 2;
        box-shadow: 0 2px 12px 0 rgba(0, 0, 0, 0.1);
}

.dark .menu_panel .inner_contaner .menu_item{
        height: 100px;
        margin: 0 auto;
        width: 100px;
        cursor: pointer;
}
```

```
.dark .menu_panel .inner_contaner .menu_item .icon{
	text-align: center;
	height: 70px;
	line-height: 70px;
}


.dark .menu_panel .inner_contaner .menu_item .text{
	text-align: center;
	height: 30px;
	line-height: 30px;
}

.dark .menu_panel .inner_contaner .menu_item:hover{
}

.dark .menu_panel .inner_contaner .menu_item:hover .icon,.dark .menu_panel .inner_contaner .
menu_item:hover .text{
	color:#2891ff
}

.dark .menu_panel .inner_contaner .menuPanelCloseBtn{
	position: absolute;
	right: -10px;
	top: -10px;
	cursor: pointer;
/*	width:30px;
	height:30px;
	border-radius: 100%;
	background: #409EFF; */
}
.dark .menu_panel .inner_contaner .menuPanelCloseBtn:hover{
	color:#F56C6C
}


.dark .setting_panel{
}

.dark .setting_panel .setting_panel_maker{
	position: fixed;
	left: 0px;
	top: 0px;
	width: 100%;
```

```
        height: 100%;
        z-index: 1;
}

.dark .setting_panel .inner_contaner{
        position: fixed;
        background: #f56c6c47;
        border-radius: 5px;
        z-index: 2;
        box-shadow: 0 2px 12px 0 rgba(0, 0, 0, 0.1);
}


.dark .setting_panel .inner_contaner .settingPanelCloseBtn{
        position: absolute;
        right: -10px;
        top: -10px;
        cursor: pointer;
}
.dark .setting_panel .inner_contaner .settingPanelCloseBtn:hover{
        color:#F56C6C
}

.dark .setting_panel .inner_contaner .setting_panel_title{
        height: 60px;
        line-height: 60px;
        text-align: center;
        font-size: 20px;
}
.dark .setting_panel .inner_contaner .settingForm {
        padding-right: 70px;
}
.dark .setting_panel .inner_contaner .settingForm .el-form-item__label{
        color: #fff;
}
```

- Public/bluc_theme.css：实现蓝色主题相关样式的定义实现。

```
/**
 * 蓝色主题
 */
@charset "utf-8";

html,body{
        margin: 0;
        padding: 0;
```

```
        width: 100%;
        /* background: #0F2B5E; */
        color: #fff;
        overflow: hidden;
}

.blue{
        position: fixed;
        left: 0;
        top:0;
        width: 100%;
        height: 100%;
        background: #0f197f;
}

.blue .menu_btn{
        position: fixed;
        right: 1%;
        bottom: 20px;
        width: 60px;
        color: #000000;
}
.blue .menu_btn_item{
        height: 60px;
        line-height: 60px;
        text-align: center;
        background: rgba(255,255,255,1);
        cursor: pointer;
}

.blue .menu_btn_item:hover{
        background: #F56C6C;
        color:#fff;
}
.blue .title{
        text-align: center;
        color: #fff;
        font-size: 24px;
        /* border-bottom: 1px solid #f56c6c47; */
}

.blue .menu_panel{
}
```

```
.blue .menu_panel .menu_panel_maker{
	position: fixed;
	left: 0px;
	top: 0px;
	width: 100%;
	height: 100%;
	z-index: 1;
}

.blue .menu_panel .inner_contaner{
	position: fixed;
	background: #f56c6c47;
	border-radius: 5px;
	z-index: 2;
	box-shadow: 0 2px 12px 0 rgba(0, 0, 0, 0.1);
}

.blue .menu_panel .inner_contaner .menu_item{
	height: 100px;
	margin: 0 auto;
	width: 100px;
	cursor: pointer;
}


.blue .menu_panel .inner_contaner .menu_item .icon{
	text-align: center;
	height: 70px;
	line-height: 70px;
}


.blue .menu_panel .inner_contaner .menu_item .text{
	text-align: center;
	height: 30px;
	line-height: 30px;
}

.blue .menu_panel .inner_contaner .menu_item:hover{
}

.blue .menu_panel .inner_contaner .menu_item:hover .icon,.blue .menu_panel .inner_contaner .
menu_item:hover .text{
	color:#2891ff
```

```
}

.blue .menu_panel .inner_contaner .menuPanelCloseBtn{
	position: absolute;
	right: -10px;
	top: -10px;
	cursor: pointer;
/*	width:30px;
	height:30px;
	border-radius: 100%;
	background: #409EFF; */
}
.blue .menu_panel .inner_contaner .menuPanelCloseBtn:hover{
	color:#F56C6C
}


.blue .setting_panel{
}

.blue .setting_panel .setting_panel_maker{
	position: fixed;
	left: 0px;
	top: 0px;
	width: 100%;
	height: 100%;
	z-index: 1;
}

.blue .setting_panel .inner_contaner{
	position: fixed;
	background: #f56c6c47;
	border-radius: 5px;
	z-index: 2;
	box-shadow: 0 2px 12px 0 rgba(0, 0, 0, 0.1);
}


.blue .setting_panel .inner_contaner .settingPanelCloseBtn{
	position: absolute;
	right: -10px;
	top: -10px;
	cursor: pointer;
}
```

```
.blue .setting_panel .inner_contaner .settingPanelCloseBtn:hover{
	color:#F56C6C
}

.blue .setting_panel .inner_contaner .setting_panel_title{
	height: 60px;
	line-height: 60px;
	text-align: center;
	font-size: 20px;
}
.blue .setting_panel .inner_contaner .settingForm {
	padding-right: 70px;
}
.blue .setting_panel .inner_contaner .settingForm .el-form-item__label{
	color: #fff;
}
```

2. 工具方法：

文件：public/utils.js

说明：界面工具方法，包括加载动画、工具方法等。

代码：

```
export default {
	name:"utils",
	/**
	 * 保存界面配置参数
	 * @param {Object} config
	 */
	saveConfig:function(config){
		localStorage.setItem("config",JSON.stringify(config));
	},
	/**
	 * 创建图表标题
	 * @param {Object} title
	 */
	createChartTitle:function(title){
		return {
			text: title,
			textStyle: {
				color: "#fff"
			},
			x: 'center',
```

```
            y: '10'
        }
    },
    /**
     * 创建图表背景
     * @param {Object} title
     */
    createChartGaid:function(left,right,top,bottom){
        return {
            left:left?left:'30',
            right:right?right:'10',
            top:top?top:'50px',
            bottom:bottom?bottom:'40'
        }
    },
    /**
     * 创建图表背景
     * @param {Object} title
     */
    createChartBaseOption:function(title,left,right,top,bottom,categorys){
        return {
            title: this.createChartTitle(title),
            tooltip: {
                show: true,
                trigger: 'axis'
            },
            grid:this.createChartGaid(left,right,top,bottom),
            xAxis: {
                type: 'category',
                axisLine: {
                    show: true,
                    lineStyle: {
                        color: this.getChartXColor()
                    }
                },
                axisLabel: {
                    color: this.getChartXTextColor()
                },
                axisTick: {
                    show: false
                },
                splitLine: {
                    show: false
                },
```

```
                boundaryGap: false,
                data: categorys
            },
            yAxis: {
                type: 'value',
                axisLabel: {
                    color: this.getChartYTextColor(),
                },
                axisLine: {
                    show: true,
                    lineStyle: {
                        color: this.getChartXColor()
                    }
                },
                splitLine: {
                    lineStyle: {
                        color: this.getChartYColor(),
                        type: 'dashed'
                    },
                }
            }
        }
    },
    /**
     * 获取 x 轴颜色
     * @param {Object} title
     */
    getChartXColor:function(){
        return '#85C1D9'
        // return '#F56C6C'
    },
    /**
     * 获取 x 轴文本颜色
     * @param {Object} title
     */
    getChartXTextColor:function(){
        // return '#8BC4F2'
        return '#fff'
    },
    /**
     * 获取 y 轴颜色
     * @param {Object} title
     */
    getChartYColor:function(){
```

```
            return '#355C84'
            // return '#F56C6C'
        },
        /**
         *  获取 y 轴文本颜色
         * @param {Object} title
         */
        getChartYTextColor:function(){
            // return '#8BC4F2'
            return '#fff'
        }
}

Date.prototype.format = function(fmt) {
        var o = {
            "M+" : this.getMonth()+1,                 //月份
            "d+" : this.getDate(),                    //日
            "h+" : this.getHours(),                   //小时
            "m+" : this.getMinutes(),                 //分
            "s+" : this.getSeconds(),                 //秒
            "q+" : Math.floor((this.getMonth()+3)/3), //季度
            "S"  : this.getMilliseconds()             //毫秒
        };
        if(/(y+)/.test(fmt)) {
                fmt=fmt.replace(RegExp.$1, (this.getFullYear()+"").substr(4 - RegExp.$1.length));
        }
        for(var k in o) {
            if(new RegExp("("+ k +")").test(fmt)){
                fmt = fmt.replace(RegExp.$1, (RegExp.$1.length==1) ? (o[k]) : (("00"+
o[k]).substr((""+ o[k]).length)));
            }
        }
        return fmt;
}
```

3. 接口方法：

文件：public/api.js

说明：系统数据加载方法，定义数据加载地址、路径及可调用的加载方法等。

代码：

```
import axios from 'axios';
const config = {
```

```
    // baseURL: process.env.baseURL
    baseURL: 'http://localhost:3000',
    timeout: 1000,
    headers: {
        'Content-Type': 'application/json',
    },
};
const api = axios.create(config);

// 默认 post 请求，使用 application/json 形式
api.defaults.headers.post['Content-Type'] = 'application/json';

/**
 * 接口地址配置
 */
const server_urls = {
    //cpu 数据加载地址
    cpu_url: '/src/assets/data/cpu.json',
    //内存数据加载地址
    memory_url: '/src/assets/data/memory.json',
    //磁盘数据加载地址
    dist_url: '/src/assets/data/dist.json',
    //网络数据加载地址
    net_url: '/src/assets/data/net.json',
    //浏览量数据加载地址
    view_url: '/src/assets/data/view.json',
    //磁盘数据加载地址
    dist_url: '/src/assets/data/dist.json',
    //数据库数据加载地址
    database_url: '/src/assets/data/database.json',
    //告警数据加载地址
    alert_url: '/src/assets/data/alert.json'
};

export default {
    name: "api",
    /**
     * 加载 CPU 数据
     * @param {Object} params
     * @param {Object} callback
     */
    loadCpuData: function(params, callback) {
        axios({
            method: 'post',
```

```
            url: config.baseURL + server_urls.cpu_url,
            params,
            headers: {
                'Content-Type': 'application/json; charset=utf-8',
            },
        }).then(response => {
            callback && callback(response);
        })
    },
    /**
     * 加載內存数据
     * @param {Object} params
     * @param {Object} callback
     */
    loadMemoryData: function(params, callback) {
        axios({
            method: 'post',
            url: config.baseURL + server_urls.memory_url,
            params,
            headers: {
                'Content-Type': 'application/json; charset=utf-8',
            },
        }).then(response => {
            callback && callback(response);
        })
    },
    /**
     * 加載网络数据
     * @param {Object} params
     * @param {Object} callback
     */
    loadNetData: function(params, callback) {
        axios({
            method: 'post',
            url: config.baseURL + server_urls.net_url,
            params,
            headers: {
                'Content-Type': 'application/json; charset=utf-8',
            },
        }).then(response => {
            callback && callback(response);
        })
    },
    /**
```

```
    * 加載浏览量数据
    * @param {Object} params
    * @param {Object} callback
    */
    loadViewData: function(params, callback) {
        axios({
            method: 'post',
            url: config.baseURL + server_urls.view_url,
            params,
            headers: {
                'Content-Type': 'application/json; charset=utf-8',
            },
        }).then(response => {
            callback && callback(response);
        })
    },
    /**
    * 加載磁盘数据
    * @param {Object} params
    * @param {Object} callback
    */
    loadDistData: function(params, callback) {
        axios({
            method: 'post',
            url: config.baseURL + server_urls.dist_url,
            params,
            headers: {
                'Content-Type': 'application/json; charset=utf-8',
            },
        }).then(response => {
            callback && callback(response);
        })
    },
    /**
    * 加載数据库数据
    * @param {Object} params
    * @param {Object} callback
    */
    loadDatabaseData: function(params, callback) {
        axios({
            method: 'post',
            url: config.baseURL + server_urls.database_url,
            params,
            headers: {
```

```
                    'Content-Type': 'application/json; charset=utf-8',
                },
            }).then(response => {
                callback && callback(response);
            })
        },
        /**
         * 加載告警数据
         * @param {Object} params
         * @param {Object} callback
         */
        loadAlertData: function(params, callback) {
            axios({
                method: 'post',
                url: config.baseURL + server_urls.alert_url,
                params,
                headers: {
                    'Content-Type': 'application/json; charset=utf-8',
                },
            }).then(response => {
                callback && callback(response);
            })
        }
}
```

4. 配置管理：

文件：public/config.js

说明：界面配置项管理文件，定义默认配置数据，配置项目加载保存等。

代码：

```
export default {
    defaultconfig:[{
        key:'theme',
        value:'dark',
        label:'界面主题',
        type:'select',
        required:true,
        options:[{
            value:'dark',
            label:'深色主题'
        },{
            value:'blue',
```

```
                label:'蓝色主题'
            }]
        },{
            key:'refreshtime',
            value:10*1000,
            label:'刷新时间',
            type:'number',
            required:true
        },{
            key:'turntime',
            value:20*1000,
            label:'切换时间',
            type:'select',
            required:true,
            options:[{
                value:20*1000,
                label:'20 秒'
            },{
                value:60*1000,
                label:'1 分钟'
            },{
                value:2*60*1000,
                label:'2 分钟'
            },{
                value:5*60*1000,
                label:'5 分钟'
            }]
        },{
            key:'serverurl',
            value:'/',
            label:'接口地址',
            type:'text',
            required:true
        }],
        /**
         * 获取系统配置
         */
        getConfig:function(){
            var config = localStorage.getItem("config");
            if(!config){
                config = this.defaultconfig;
            }else{
                config = JSON.parse(config);
            }
```

```
        return config;
    }
}
```

5. 数据文件：

文件：data/*

说明：使用 json 格式模拟数据，实际中请直接调用后台真实接口。

- alert.json：告警相关数据模拟

```
{
    "types": [{
        "value": "1",
        "label":"CPU 超频"
    },{
        "value": "2",
        "label":"内存超額"
    },{
        "value": "3",
        "label":"网络上行超限"
    },{
        "value": "4",
        "label":"网络下行超限"
    },{
        "value": "5",
        "label":"访问超限"
    },{
        "value": "6",
        "label":"数据库超限"
    }],
    "status": [{
        "value": "0",
        "label":"未处理"
    },{
        "value": "1",
        "label":"已处理"
    }],
    "leavel": [{
        "value": "1",
        "label":"提示"
    },{
        "value": "2",
        "label":"警告"
```

```
    },{
        "value": "3",
        "label":"重要 "
    },{
        "value": "4",
        "label":"致命"
    }],
    "values": [{
        "value": 40,
        "type": "1",
        "status": "0",
        "leavel": "1",
        "from": "cpu"
    },{
        "value": 40,
        "type": "2",
        "status": "1",
        "leavel": "1",
        "from": "cpu"
    },{
        "value": 40,
        "type": "3",
        "status": "0",
        "leavel": "2",
        "from": "cpu"
    },{
        "value": 40,
        "type": "4",
        "status": "1",
        "leavel": "1",
        "from": "cpu"
    },{
        "value": 40,
        "type": "1",
        "status": "1",
        "leavel": "1",
        "from": "memory"
    },{
        "value": 40,
        "type": "2",
        "status": "0",
        "leavel": "1",
        "from": "memory"
    },{
```

```
        "value": 22,
        "type": "3",
        "status": "1",
        "leavel": "2",
        "from": "memory"
    },{
        "value": 40,
        "type": "4",
        "status": "1",
        "leavel": "3",
        "from": "memory"
    },{
        "value": 40,
        "type": "6",
        "status": "1",
        "leavel": "4",
        "from": "netin"
    },{
        "value": 40,
        "type": "6",
        "status": "0",
        "leavel": "3",
        "from": "netin"
    },{
        "value": 22,
        "type": "3",
        "status": "1",
        "leavel": "2",
        "from": "netin"
    },{
        "value": 40,
        "type": "4",
        "status": "1",
        "leavel": "4",
        "from": "netin"
    },{
        "value": 40,
        "type": "5",
        "status": "1",
        "leavel": "1",
        "from": "netout"
    },{
        "value": 40,
        "type": "2",
```

```
            "status": "1",
            "leavel": "4",
            "from": "netout"
        },{
            "value": 22,
            "type": "3",
            "status": "5",
            "leavel": "1",
            "from": "netout"
        },{
            "value": 40,
            "type": "4",
            "status": "1",
            "leavel": "2",
            "from": "netout"
        }]
}
```

- cpu.json：cpu 相关数据模拟

```
{
    "curdata": 65,
    "curdatas":[{
        "name":"CPU1",
        "value":15
    },{
        "name":"CPU2",
        "value":15
    },{
        "name":"CPU3",
        "value":15
    },{
        "name":"CPU4",
        "value":15
    }],
    "cpus":["CPU1","CPU2","CPU3","CPU4"],
    "records": {
        "times":["21:56", "21:57", "21:58", "21:59", "22:00", "22:01", "22:02", "22:03", "22:04",
"22:05", "22:06",
        "22:07", "22:08", "22:09", "22:10", "22:11", "22:12", "22:13", "22:14", "22:15", "22:16",
"22:17", "22:18",
        "22:19", "22:20", "22:21", "22:22", "22:23", "22:24", "22:25"],
        "CPU1":["64.0", "97.9", "59", "97.1", "33.1", "74", "65.5", "56.1", "72.7", "39.4", "98.0",
"91.8", "47.8", "27.3", "30",
        "4", "26.0", "22.5", "21", "14.1", "73.6", "23.0", "26.9", "29.1", "41.2", "12.8", "99.5",
"75.4", "66.2", "14.9"],
```

```
            "CPU2":["0.11", "0.78", "0.57", "0.41", "0.79", "0.96", "0.52", "0.29", "0.23", "0.53",
"0.21", "0.77", "0.68",
            "0.58", "1.00", "0.54", "0.49", "0.52", "0.93", "0.74", "0.29", "0.80", "0.45", "0.81",
"0.21", "0.49",
            "0.91", "0.66", "0.27", "0.17", "0.56"],
            "CPU3":["0.34", "0.25", "0.49", "0.97", "0.09", "0.47", "0.28", "0.61", "0.57", "0.48",
"0.38", "0.66", "0.96",
            "0.06", "0.60", "0.02", "0.08", "0.98", "0.15", "0.04", "0.31", "0.93", "0.41", "0.51",
"0.49", "0.37",
            "0.94", "0.07", "0.93", "0.99", "0.40"]
      }
}
```

- ❖ database.json：数据库相关数据模拟

```
{
      "types": [{
            "value": "1",
            "name": "使用中"
      }, {
            "name": "执行中",
            "value": "2"
      }, {
            "name": "空闲中",
            "value": "3"
      }, {
            "name": "新链接",
            "value": "4"
      }, {
            "name": "关闭中",
            "value": "5"
      }],
      "max": 1000,
      "curdata": [{
            "value": 1,
            "type": "1"
      }, {
            "type": "2",
            "value": 2
      }, {
            "type": "3",
            "value": 3
      }, {
            "type": "4",
            "value": 4
      }, {
```

```
            "type": "5",
            "value": 5
        }],
        "records": {
            "times":["21:56", "21:57", "21:58", "21:59", "22:00", "22:01", "22:02", "22:03", "22:04",
"22:05", "22:06",
            "22:07", "22:08", "22:09", "22:10", "22:11", "22:12", "22:13", "22:14", "22:15", "22:16",
"22:17", "22:18",
            "22:19", "22:20", "22:21", "22:22", "22:23", "22:24", "22:25"],
            "_1":["64.0", "97.9", "59", "97.1", "33.1", "74", "65.5", "56.1", "72.7", "39.4", "98.0",
"91.8", "47.8", "27.3", "30",
            "4", "26.0", "22.5", "21", "14.1", "73.6", "23.0", "26.9", "29.1", "41.2", "12.8", "99.5",
"75.4", "66.2", "14.9"],
            "_2":["0.11", "0.78", "0.57", "0.41", "0.79", "0.96", "0.52", "0.29", "0.23", "0.53",
"0.21", "0.77", "0.68",
            "0.58", "1.00", "0.54", "0.49", "0.52", "0.93", "0.74", "0.29", "0.80", "0.45", "0.81",
"0.21", "0.49",
            "0.91", "0.66", "0.27", "0.17", "0.56"],
            "_3":["0.34", "0.25", "0.49", "0.97", "0.09", "0.47", "0.28", "0.61", "0.57", "0.48",
"0.38", "0.66", "0.96",
            "0.06", "0.60", "0.02", "0.08", "0.98", "0.15", "0.04", "0.31", "0.93", "0.41", "0.51",
"0.49", "0.37",
            "0.94", "0.07", "0.93", "0.99", "0.40"],
            "_4":["0.34", "0.25", "0.49", "0.97", "0.09", "0.47", "0.28", "0.61", "0.57", "0.48",
"0.38", "0.66", "0.96",
            "0.06", "0.60", "0.02", "0.08", "0.98", "0.15", "0.04", "0.31", "0.93", "0.41", "0.51",
"0.49", "0.37",
            "0.94", "0.07", "0.93", "0.99", "0.40"],
            "_5":["0.34", "0.25", "0.49", "0.97", "0.09", "0.47", "0.28", "0.61", "0.57", "0.48",
"0.38", "0.66", "0.96",
            "0.06", "0.60", "0.02", "0.08", "0.98", "0.15", "0.04", "0.31", "0.93", "0.41", "0.51",
"0.49", "0.37",
            "0.94", "0.07", "0.93", "0.99", "0.40"]
        }
}
```


- dist.json：磁盘相关数据模拟


```
[{
        "name": "C 盘",
        "value": 20,
        "all":1024
}, {
        "name": "D 盘",
        "value": 37.8,
        "all":2048
```

```
}, {
    "name": "E 盘",
    "value": 41,
    "all":1024
}, {
    "name": "F 盘",
    "value": 50,
    "all":500
}, {
    "name": "G 盘",
    "value": 66,
    "all":4086
}]
```

- memory.json：内存相关数据模拟

```
{
    "curdata": 65,
    "records": {
        "times":["21:56", "21:57", "21:58", "21:59", "22:00", "22:01", "22:02", "22:03", "22:04",
"22:05", "22:06",
        "22:07", "22:08", "22:09", "22:10", "22:11", "22:12", "22:13", "22:14", "22:15", "22:16",
"22:17", "22:18",
        "22:19", "22:20", "22:21", "22:22", "22:23", "22:24", "22:25"],
        "values":["64.0", "97.9", "59", "97.1", "33.1", "74", "65.5", "56.1", "72.7", "39.4",
"98.0", "91.8", "47.8", "27.3", "30",
        "4", "26.0", "22.5", "21", "14.1", "73.6", "23.0", "26.9", "29.1", "41.2", "12.8", "99.5",
"75.4", "66.2", "14.9"]
    }
}
```

- net.json：网络相关数据模拟

```
{
    "times": ["21:51", "21:52", "21:53", "21:54", "21:55", "21:56", "21:57", "21:58", "21:59",
"22:00", "22:01",
        "22:02", "22:03", "22:04", "22:05", "22:06", "22:07", "22:08", "22:09", "22:10", "22:11",
"22:12", "22:13",
        "22:14", "22:15", "22:16", "22:17", "22:18", "22:19", "22:20", "22:21"
    ],
    "ins": ["0.11", "0.78", "0.57", "0.41", "0.79", "0.96", "0.52", "0.29", "0.23", "0.53", "0.21",
"0.77", "0.68",
        "0.58", "1.00", "0.54", "0.49", "0.52", "0.93", "0.74", "0.29", "0.80", "0.45", "0.81",
"0.21", "0.49",
        "0.91", "0.66", "0.27", "0.17", "0.56"
    ],
    "outs": ["0.34", "0.25", "0.49", "0.97", "0.09", "0.47", "0.28", "0.61", "0.57", "0.48", "0.38",
```

```
"0.66", "0.96",
        "0.06", "0.60", "0.02", "0.08", "0.98", "0.15", "0.04", "0.31", "0.93", "0.41", "0.51",
"0.49", "0.37",
        "0.94", "0.07", "0.93", "0.99", "0.40"
    ]
}
```

- view.json：访问情况相关数据模拟

```
{
    "clients": [{
        "value": "1",
        "label":"PC"
    },{
        "value": "2",
        "label":"移动端"
    },{
        "value": "3",
        "label":"微信"
    },{
        "value": "4",
        "label":"浏览器"
    }],
    "urls": [{
        "value": "1",
        "label":"个人信息"
    },{
        "value": "2",
        "label":"登录"
    },{
        "value": "3",
        "label":"修改密码"
    },{
        "value": "4",
        "label":"资产查询"
    },{
        "value": "5",
        "label":"商品查询"
    }],
    "times": ["21:56", "21:57", "21:58", "21:59", "22:00", "22:01", "22:02", "22:03", "22:04",
"22:05", "22:06",
        "22:07", "22:08", "22:09", "22:10", "22:11", "22:12", "22:13", "22:14", "22:15", "22:16",
"22:17", "22:18",
        "22:19", "22:20", "22:21", "22:22", "22:23", "22:24", "22:25"
    ],
    "values": ["640", "979", "59", "971", "331", "74", "655", "561", "727", "394", "980", "918",
```

```
"478", "273", "30",
            "4", "260", "225", "21", "141", "736", "230", "269", "291", "412", "128", "995", "754",
"662", "149"
      ],
      "record": [{
            "value": 40,
            "clients": "1",
            "url": "1"
      },{
            "value": 40,
            "clients": "2",
            "url": "2"
      },{
            "value": 40,
            "clients": "2",
            "url": "2"
      },{
            "value": 40,
            "clients": "3",
            "url": "3"
      },{
            "value": 40,
            "clients": "4",
            "url": "4"
      },{
            "value": 40,
            "clients": "3",
            "url": "5"
      },{
            "value": 22,
            "clients": "2",
            "url": "2"
      },{
            "value": 40,
            "clients": "1",
            "url": "1"
      },{
            "value": 40,
            "clients": "1",
            "url": "2"
      },{
            "value": 40,
            "clients": "1",
            "url": "2"
```

```
    },{
        "value": 22,
        "clients": "1",
        "url": "3"
    },{
        "value": 40,
        "clients": "1",
        "url": "1"
    }]
}
```

8.3.2 展示配置界面实现

1. 配置说明：

通过此配置界面配置监控界面的一下参数，包括主题、布局、数据刷新周期等各种参数。配置后保存到浏览器 Storage 中，使用 JSON 格式进行保存。

默认配置参数如下：

```
[{
        key:'theme',
        value:'dark',
        label:'界面主题',
        type:'select',
        required:true,
        options:[{
            value:'dark',
            label:'深色主题'
        },{
            value:'blue',
            label:'蓝色主题'
        }]
    },{
        key:'refreshtime',
        value:10*1000,
        label:'刷新时间',
        type:'number',
        required:true
    },{
        key:'turntime',
        value:20*1000,
        label:'切换时间',
```

```
            type:'select',
            required:true,
            options:[{
                value:20*1000,
                label:'20 秒'
            },{
                value:60*1000,
                label:'1 分钟'
            },{
                value:2*60*1000,
                label:'2 分钟'
            },{
                value:5*60*1000,
                label:'5 分钟'
            }]
        },{
            key:'serverurl',
            value:'/',
            label:'接口地址',
            type:'text',
            required:true
        }]
```

2. 配置项目：

序号	配置项	配置说明
1	主题配置	1.深色主题　　2.深蓝主题
2	刷新周期配置	1 分钟　5 分钟　10 分钟等
3	服务器路径	数据加载服务器地址
4	切换周期	菜单自动切换周期：1 分钟、5 分钟、15 分钟等

3. 实现代码：

```
/**
 * 恢复默认
 */
defalutForm: function() {
    var that = this;
    this.configs = config.defaultconfig;
    this.configs.forEach(function(item, index, arr) {
```

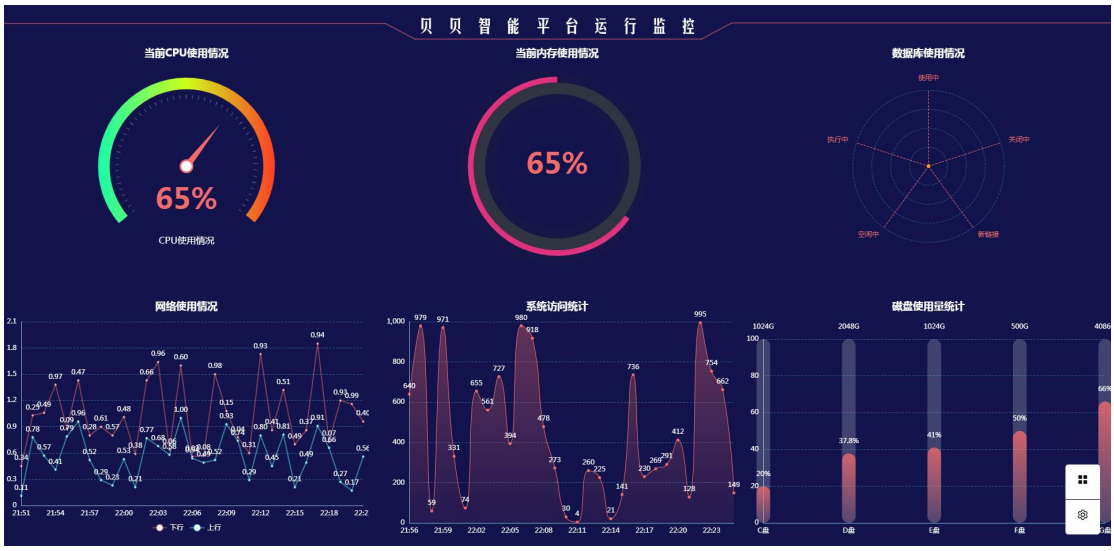
```
            that.settingForm[item.key] = item.value;

            that.rules[item.key] = [{
                required: item.required,
                message: '请输入' + item.label,
                trigger: 'blur'
            }];
        });
    },
    /**
     * 提交表单
     */
    submitForm: function() {
        var that = this;
        this.$refs.settingFormRef.validate((valid, fields) => {
            if (!valid) {
                return;
            }
            //将数据组合成数组格式
            var configdata = [];
            for (var key in that.settingForm) {
                var itemconfig = null;
                for (var i = 0; i < that.configs.length; i++) {
                    if (that.configs[i].key == key) {
                        itemconfig = that.configs[i];
                        break;
                    }
                }
                if (!itemconfig) { //未识别的配置项
                    continue;
                }
                //更新数据
                itemconfig.value = that.settingForm[key];
                configdata.push(itemconfig);
            }

            //保存系统配置
            utils.saveConfig(configdata);

            //刷新并应用系统配置
            this.refreshConfig();
        });
    },
    /**
```

```
 * 刷新并应用系统配置
 */
refreshConfig: function() {
    //更新配置表单对应数据
    this.configs = config.getConfig();
    var that = this;
    this.configs.forEach(function(item, index, arr) {
        that.settingForm[item.key] = item.value;

        that.rules[item.key] = [{
            required: item.required,
            message: '请输入' + item.label,
            trigger: 'blur'
        }];

        //更新界面主题
        if (item.key == 'theme') {
            that.curTheme = item.value
        }

        //简单期间，直接刷新界面
        window.location.reload();
    });
}
<!-- 系统配置弹框 -->
<div class="setting_panel" v-if="settingshow">
    <div class="setting_panel_maker" @click="hideSetting">
        <div class="inner_contaner" @click.stop="stopPropagation">
            <el-icon :size="25" class="settingPanelCloseBtn" @click="hideSetting">
                <Close-bold :size="25" />
            </el-icon>
            <div class="setting_panel_title">
                系统配置
            </div>
            <el-form ref="settingFormRef" :model="settingForm" status-icon :rules="rules"
label-width="120px"
                class="settingForm">
                <el-form-item :label="item.label+':'" prop="item.type" v-for="(item,index) in
configs">
                    <!-- 识别配置项类型并创建对应组件 -->
                    <el-input v-if="item.type != 'select'" v-model="settingForm[item.key]"
                        placeholder="请输入{{item.label}}" type="item.type"
autocomplete="off" />
                    <el-select v-if="item.type == 'select'" style="width:100%"
```

```
v-model="settingForm[item.key]"
                                placeholder="请选择{{item.label}}">
                                <el-option                v-for="(optionitem,index1)                in
item.options" :label="optionitem.label"
                                    :value="optionitem.value" />
                            </el-select>
                        </el-form-item>
                        <el-form-item>
                            <el-row style="width:100%">
                                <el-col :span="11">
                                    <el-button          style="width:100%"          type="primary"
@click="submitForm(settingFormRef)">
                                        保存配置</el-button>
                                </el-col>
                                <el-col :span="2">
                                </el-col>
                                <el-col :span="11">
                                    <el-button                                    style="width:100%"
@click="defalutForm(settingFormRef)">恢复默认</el-button>
                                </el-col>
                            </el-row>
                        </el-form-item>
                    </el-form>

                </div>
            </div>
        </div>
```

8.3.3 综合监控界面实现

1. 效果预览：



2. 文件路径：

/pages/Index.vue

3. 模板代码：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col :span="8">
            <div class="grid-content bg-purple chart_panel" id="cpuchart">cpu 使用量
            </div>
        </el-col>
        <el-col :span="8">
            <div class="grid-content bg-purple chart_panel" id="memorychart">内存使用量
            </div>
        </el-col>
        <el-col :span="8">
            <div class="grid-content bg-purple chart_panel" id="databasehart">数据库链接
状况
```

```
                </div>
            </el-col>
        </el-row>
        <el-row :gutter="10">
            <el-col :span="8">
                <div class="grid-content bg-purple chart_panel" id="netchart">网络使用情况
                </div>
            </el-col>
            <el-col :span="8">
                <div class="grid-content bg-purple chart_panel" id="viewchart">当前访问量
                </div>
            </el-col>
            <el-col :span="8">
                <div class="grid-content bg-purple chart_panel" id="distchart">磁盘使用量
                </div>
            </el-col>
        </el-row>
</template>
```

4. CSS 样式：

```
<style scoped="scoped">
    .title {
        height: v-bind(titleheight+"px");
        line-height: v-bind(titleheight+"px");
    }

    .chart_panel {
        height: v-bind(chartheight+"px");
    }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
    import utils from '../public/utils.js'
    import api from '../public/api.js'
    import config from '../public/config.js'

    import * as echarts from 'echarts'

    export default {
```

```
data() {
    return {
        chartheight: 100,
        titleheight: 60,
        timer: null,
        charts: []
    }
},
mounted: function() {
    this.chartheight = ($(window).height() - this.titleheight) / 2;
    this.loadData();

    //开始定时刷新报表数据
    this.startRefreshChart();

    var that = this;
    $(window).resize(function() {
        this.chartheight = ($(window).height() - this.titleheight) / 2;
        for (var key in that.charts) {
            that.charts[key].resize();
        }
    });
},
unmounted: function() {
    if (this.timer) {
        clearInterval(this.timer);
    }
},
methods: {
    /**
     * 定时刷新报表数据
     */
    startRefreshChart: function() {
        if (this.timer) {
            clearInterval(this.timer);
        }
        var that = this;

        //获取刷新周期，TODO 配置变动时，此处需自动更新
        var refreshtime = 60 * 1000;
        config.getConfig().forEach(function(item, index) {
            if (item.key == 'refreshtime') {
                refreshtime = item.value;
            }
```

```
                });

                this.timer = setInterval(function() {
                    //刷新 cpu 数据
                    var cpudata = (Math.random() * 100).toFixed(2);
                    var option = that.charts['cpu'].getOption();
                    option.series[0].data[0].value = cpudata;
                    that.charts['cpu'].setOption(option);

                    //刷新内存数据
                    var memorydata = (Math.random() * 100).toFixed(2);
                    var option = that.charts['memory'].getOption();
                    option.series[0].data[0].value = memorydata;
                    option.series[0].data[1].value = 100 - memorydata;
                    option.title[1].text = memorydata + '%';
                    that.charts['memory'].setOption(option);

                    //更新数据库使用情况
                    var option = that.charts['database'].getOption();
                    var databasedata = [(Math.random() * 1000).toFixed(0), (Math.random()
* 1000).toFixed(0), (
                        Math.random() * 1000).toFixed(0), (Math.random() *
1000).toFixed(0), (Math
                        .random() * 1000).toFixed(0)];
                    option.series[0].data[0] = databasedata;
                    that.charts['database'].setOption(option);

                    //更新网络使用情况
                    var option = that.charts['net'].getOption();
                    var times = [];
                    var indata = [];
                    var outdata = [];
                    var startData = new Date();
                    startData.setMinutes(startData.getMinutes() - 30);
                    for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                            1)) {
                        times.push(i.format('hh:mm'));
                        indata.push((Math.random(100) * 100).toFixed(2));
                        outdata.push((Math.random(100) * 100).toFixed(2));
                    }
                    option.xAxis[0].data = times;
                    option.series[0].data = indata;
                    option.series[1].data = outdata;
```

```
                    that.charts['net'].setOption(option);

                    //更新访问情况
                    var option = that.charts['view'].getOption();
                    var startData = new Date();
                    var times = [];
                    var values = [];
                    startData.setMinutes(startData.getMinutes() - 30);
                    for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                            1)) {
                        times.push(i.format('hh:mm'));
                        values.push((Math.random(1000) * 1000).toFixed(0));
                    }
                    option.xAxis[0].data = times;
                    option.series[0].data = values;
                    that.charts['view'].setOption(option);

                    //更新访问情况
                    var option = that.charts['dist'].getOption();
                    var yData = [];
                    yData.push((Math.random(100) * 100).toFixed(2));
                    yData.push((Math.random(100) * 100).toFixed(2));
                    yData.push((Math.random(100) * 100).toFixed(2));
                    yData.push((Math.random(100) * 100).toFixed(2));
                    yData.push((Math.random(100) * 100).toFixed(2));

                    option.series[0].data = yData;
                    that.charts['dist'].setOption(option);

                }, refreshtime);
            },
            /**
             * 加载数据
             */
            loadData: function() {
                var that = this;
                //加载 cpu 数据
                api.loadCpuData({}, function(res) {
                    that.initCpuChart(res);
                });
                //加载内存数据
                api.loadMemoryData({}, function(res) {
                    that.initMemoryChart(res);
```

```
        });
        //加载网络数据
        api.loadNetData({}, function(res) {
            that.initNetChart(res);
        });
        //加载浏览量数据
        api.loadViewData({}, function(res) {
            that.initViewChart(res);
        });
        //加载磁盘数据
        api.loadDistData({}, function(res) {
            that.initDistChart(res);
        });
        //加载数据库数据
        api.loadDatabaseData({}, function(res) {
            that.initDatabaseChart(res);
        });
    },
    /**
     * 初始化数据库图表
     * @param {Object} res
     */
    initDatabaseChart: function(res) {
        var myChart = this.$echarts.init(document.getElementById('databasehart'));

        var max = res.data.max;

        var types = [];
        res.data.types.forEach(function(item, index) {
            types.push({
                max: max,
                name: item.name
            });
        });

        var data = [];
        res.data.curdata.forEach(function(item, index) {
            data.push(item.value);
        });

        var option = {
            "color": ["rgba(245, 166, 35, 1)", "rgba(19, 173, 255, 1)"],
            title: utils.createChartTitle('数据库使用情况'),
            "tooltip": {
```

```
            "show": true,
            "trigger": "item"
        },
        "radar": {
            "center": ["50%", "50%"],
            "radius": "60%",
            "startAngle": 90,
            "splitNumber": 4,
            "shape": "circle",
            "splitArea": {
                "areaStyle": {
                    "color": ["transparent"]
                }
            },
            "axisLine": {
                "show": true,
                "lineStyle": {
                    color: '#f56c6c',
                    type: 'dashed'
                }
            },
            "splitLine": {
                "show": true,
                "lineStyle": {
                    color: '#355C84',
                    type: 'dashed'
                }
            },
            "indicator": types
        },
        "series": [{
            "name": "使用量",
            "type": "radar",
            "symbol": "circle",
            "symbolSize": 5,
            "areaStyle": {
                "normal": {
                    "color": "rgba(245, 166, 35, 0.4)"
                }
            },
            itemStyle: {
                color: 'rgba(245, 166, 35, 1)',
                borderColor: 'rgba(245, 166, 35, 0.3)',
                borderWidth: 1,
```

```
                                },
                                "lineStyle": {
                                    "normal": {
                                        "type": "dashed",
                                        "color": "rgba(245, 166, 35, 1)",
                                        "width": 1
                                    }
                                },
                                "data": data
                            }]
                        };
                        // 使用刚指定的配置项和数据显示图表。
                        myChart.setOption(option);
                        this.charts['database'] = myChart;
                    },
                    /**
                     * 初始化磁盘图表
                     * @param {Object} res
                     */
                    initDistChart: function(res) {
                        var myChart = this.$echarts.init(document.getElementById('distchart'));
                        let xData = [],
                            yData = [],
                            all = [];
                        res.data.forEach(function(item, index) {
                            xData.push(item.name);
                            yData.push(item.value);
                            all.push({
                                "name": item.all + "G",
                                "value": 100
                            });
                        });

                        var option = utils.createChartBaseOption('磁盘使用量统计', null, null, '80px',
null,xData);

                        option.series =    [{
                            name: '使用量',
                            type: 'bar',
                            barWidth: '16%',
                            itemStyle: {
                                normal: {
                                    barBorderRadius: 30,
                                    color: new echarts.graphic.LinearGradient(
```

```
                        0, 0, 0, 1, [{
                                offset: 0,
                                color: 'rgba(245,108,108, 0.8)'
                            },
                            {
                                offset: 1,
                                color: 'rgba(245,108,108, 0.1)'
                            }
                        ]
                    )
                }
            },
            label: {
                show: true,
                position: 'top',
                distance: 15,
                formatter: '{c}%',
                color: '#fff'
            },
            data: yData,
            zlevel: 11
        }, {
            name: '总量',
            type: 'bar',
            barWidth: '16%',
            barGap: '-100%',
            data: all,
            label: {
                show: true,
                position: 'top',
                distance: 15,
                formatter: '{b}',
                color: '#fff'
            },
            itemStyle: {
                normal: {
                    barBorderRadius: 30,
                    color: 'rgba(255,255,255,0.2)'
                }
            },
            zlevel: 9
        }];
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption(option);
```

```
                this.charts['dist'] = myChart;
            },
            /**
             * 初始化浏览量图表
             * @param {Object} res
             */
            initViewChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('viewchart'));

                var times = res.data.times;
                var values = res.data.values;

                var option = utils.createChartBaseOption('系统访问统计', '60px', null, null,
null,times);
                option.series = {
                    name: '访问量',
                    type: 'line',
                    color: '#0092f6',
                    smooth: true,
                    itemStyle: {
                        normal: {
                            color: '#f56c6c',
                            lineStyle: {
                                color: '#f56c6c',
                                width: 1,
                            },
                            areaStyle: {
                                color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [{
                                    offset: 0,
                                    color: 'rgba(245,108,108, 0.5)'
                                }, {
                                    offset: 1,
                                    color: 'rgba(245,108,108, 0.1)'
                                }], false)
                            }
                        }
                    },
                    label: {
                        show: true,
                        position: 'top',
                        textStyle: {
                            color: '#fff',
                        }
                    },
```

```
                        symbol: 'circle',
                        symbolSize: 5,
                        data: values
                    };
                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['view'] = myChart;
                },
                /**
                 * 初始化网络图表
                 * @param {Object} res
                 */
                initNetChart: function(res) {
                    var myChart = this.$echarts.init(document.getElementById('netchart'));

                    var times = res.data.times;
                    var indata = res.data.ins;
                    var outdata = res.data.outs;

                    var option = utils.createChartBaseOption('网络使用情况', null, null, null,
'70px',times);
                    option.legend = {
                        show: true,
                        x: 'center',
                        bottom: '20px',
                        textStyle: {
                            color: '#fff'
                        },
                        data: ['下行', '上行']
                    };
                    option.series = [{
                        name: '上行',
                        type: 'line',
                        stack: '总量',
                        symbolSize: 3,
                        itemStyle: {
                            color: '#55eff185',
                            borderColor: '#55eff185',
                            borderWidth: 1
                        },
                        label: {
                            show: true,
                            position: 'top',
                            textStyle: {
```

```
                        color: '#fff',
                    }
                },
                data: indata
            }, {
                name: '下行',
                type: 'line',
                stack: '总量',
                symbolSize: 3,
                itemStyle: {
                    color: 'rgba(245,108,108, 0.5)',
                    borderColor: 'rgba(245,108,108, 0.5)',
                    borderWidth: 1
                },
                label: {
                    show: true,
                    position: 'top',
                    textStyle: {
                        color: '#fff',
                    }
                },
                data: outdata
            }];

            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['net'] = myChart;
        },
        /**
         * 初始化内存图表
         * @param {Object} res
         */
        initMemoryChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('memorychart'));

            var value = res.data.curdata;
            var dataStyle = {
                normal: {
                    label: {
                        show: false
                    },
                    labelLine: {
                        show: false
                    },
```

```
                shadowBlur: 40,
                shadowColor: 'rgba(40, 40, 40, 0.5)',
            }
        };
        var placeHolderStyle = {
            normal: {
                color: 'rgba(0,0,0,0)',
                label: {
                    show: false
                },
                labelLine: {
                    show: false
                }
            },
            emphasis: {
                color: 'rgba(0,0,s0,0)'
            }
        };
        var option = {
            title: [utils.createChartTitle('当前内存使用情况'), {
                text: value + '%',
                x: 'center',
                y: 'center',
                textStyle: {
                    color: "#f56c6c",
                    fontSize: 50
                }
            }],
            color: ['#FF358F', '#313443', '#313443'],
            tooltip: {
                show: true,
                formatter: "{a} <br/>{b} : {c} ({d}%)"
            },
            series: [{
                name: 'Line 1',
                type: 'pie',
                clockWise: false,
                radius: [145, 155],
                itemStyle: dataStyle,
                data: [{
                    value: value,
                    name: '01'
                }, {
                    value: 100 - value,
```

```
                        name: 'invisible',
                        itemStyle: placeHolderStyle
                    }]
                }, {
                    name: 'Line 2',
                    type: 'pie',
                    animation: false,
                    clockWise: false,
                    radius: [125, 145],
                    itemStyle: dataStyle,
                    tooltip: {
                        show: false
                    },
                    data: [{
                        value: 100,
                        name: '02',
                        itemStyle: {
                            emphasis: {
                                color: '#313443'
                            }
                        }
                    }, {
                        value: 0,
                        name: 'invisible',
                        itemStyle: placeHolderStyle
                    }]
                }]
            };

            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['memory'] = myChart;
        },
        /**
         * 初始化 cpu 图表
         * @param {Object} res
         */
        initCpuChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('cpuchart'));
            // 指定图表的配置项和数据
            var value = res.data.curdata; //此处使用定时器刷新数据，实际情况使用接口加载数据
            var option = {
                title: utils.createChartTitle('当前 CPU 使用情况'),
```

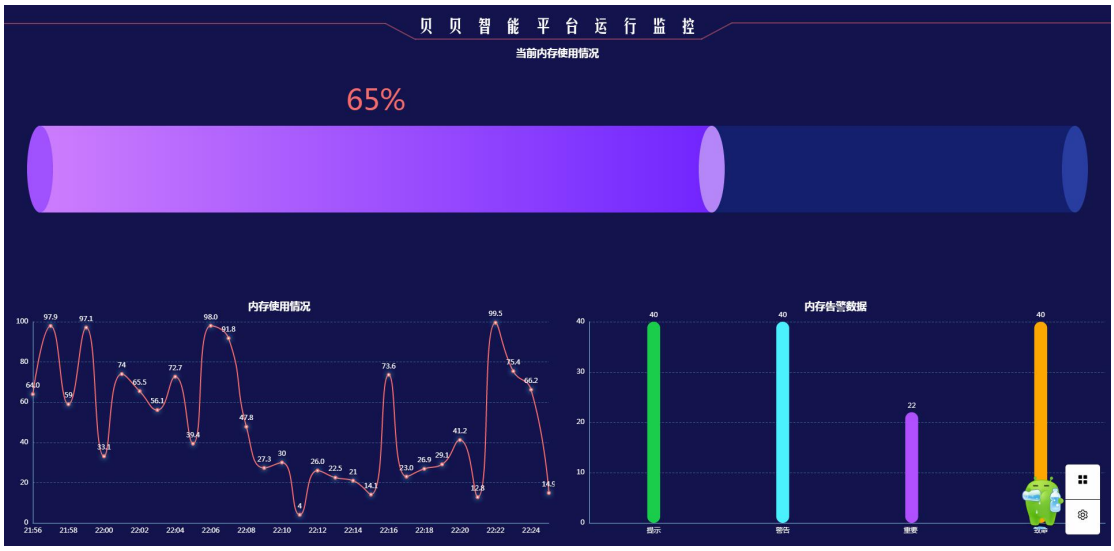
```
tooltip: {
    formatter: '{a} <br/>{b} : {c}%'
},
series: [{
    type: 'gauge',
    radius: '70%',
    startAngle: '220',
    endAngle: '-65',
    pointer: {
        show: true,
        itemStyle: {
            color: '#f56c6c'
        }
    },
    detail: {
        formatter: function(value) {
            var num = Math.round(value);
            return '{bule|' + value + '}{white|%}';
        },
        rich: {
            white: {
                fontSize: 50,
                fontWeight: 600,
                color: '#f56c6c'
            },
            bule: {
                fontSize: 50,
                fontWeight: 600,
                color: '#f56c6c'
            },
            radius: {
                width: 350,
                height: 80,
                borderWidth: 1,
                borderColor: '#0092F2',
                fontSize: 50,
                color: '#fff',
                backgroundColor: '#1B215B',
                borderRadius: 20,
                textAlign: 'center'
            },
            size: {
                height: 400,
                padding: [100, 0, 0, 0],
```

```
                        }
                    },
                    offsetCenter: ['0%', '40%']
                },
                data: [{
                    value: value,
                    name: 'CPU 使用情况',
                }],
                markPoint: {
                    data: [{
                        x: "50%",
                        y: "50%",
                        symbol: 'circle',
                        symbolSize: 24,
                        itemStyle: {
                            color: '#f56c6c'
                        },
                    }, {
                        x: "50%",
                        y: "50%",
                        symbol: 'circle',
                        symbolSize: 18,
                        itemStyle: {
                            color: "#fff"
                        },
                    }]
                },
                title: {
                    show: true,
                    color: '#fff',
                    offsetCenter: ['0', '85%'],
                    fontSize: 16
                },
                axisLine: {
                    show: true,
                    lineStyle: {
                        color: [
                            [0.91, new echarts.graphic.LinearGradient(0, 0, 1, 0,
[{
                                    offset: 0,
                                    // color: 'rgba(248,152,152,0.3)', // 0% 处
的颜色
                                    color: '#19fdab', // 0% 处的颜色
                                },
```

```
                                        {
                                            offset: 0.5,
                                            color: "#d0ff19",
                                        },
                                        {
                                            offset: 1,
                                            // color: 'rgba(245,108,108,0.9)', // 100%
处的颜色
                                            color: '#ff4026', // 100% 处的颜色
                                        },
                                    ])]
                                ],
                                width: 20,
                                shadowOffsetX: 0,
                                shadowOffsetY: 0,
                                opacity: 1
                            }
                        },
                        axisTick: {
                            show: true
                        },
                        splitLine: {
                            show: false,
                        },
                        axisLabel: {
                            show: false
                        }
                    }]
                };

                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['cpu'] = myChart;
            }
        }
    }
```

8.3.4 内存监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Memory.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="memorychart">
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="recordchart">
            </div>
        </el-col>
```

```
            <el-col :span="12">
                <div class="grid-content bg-purple chart_panel" id="alertchart">
                </div>
            </el-col>
        </el-row>
</template>
```

4. 样式控制：

```
<style scoped="scoped">
    .title {
        height: v-bind(titleheight+"px");
        line-height: v-bind(titleheight+"px");
    }

    .chart_panel {
        height: v-bind(chartheight+"px");
    }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
    import utils from '../public/utils.js'
    import api from '../public/api.js'
    import config from '../public/config.js'

    import * as echarts from 'echarts'

    export default {
        data() {
            return {
                chartheight: 100,
                titleheight: 60,
                timer: null,
                charts: []
            }
        },
        mounted: function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            this.loadData();
```

```
        //开始定时刷新报表数据
        this.startRefreshChart();

        var that = this;
        $(window).resize(function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            for (var key in that.charts) {
                that.charts[key].resize();
            }
        });
    },
    unmounted: function() {
        if (this.timer) {
            clearInterval(this.timer);
        }
    },
    methods: {
        /**
         * 定时刷新报表数据
         */
        startRefreshChart: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
            var that = this;

            //获取刷新周期，TODO 配置变动时，此处需自动更新
            var refreshtime = 60 * 1000;
            config.getConfig().forEach(function(item, index) {
                if (item.key == 'refreshtime') {
                    refreshtime = item.value;
                }
            });

            this.timer = setInterval(function() {
                //刷新 cpu 数据
                var option = that.charts['memory'].getOption();
                var value = (Math.random(100) * 100).toFixed(2);
                option.series[1].data[0].value = value;
                option.series[2].data[0] = 100 - value;
                option.series[3].data[0] = value;
                option.series[4].data[0] = 100 - value;
                that.charts['memory'].setOption(option);
```

```
                        //刷新历史数据
                        var option = that.charts['record'].getOption();

                        var times = [];
                        var curdatas = [];
                        var startData = new Date();
                        startData.setMinutes(startData.getMinutes() - 30);
                        for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                                        1)) {
                            times.push(i.format('hh:mm'));
                            curdatas.push((Math.random(100) * 100).toFixed(2));
                        }
                        option.series[0].data = curdatas;
                        that.charts['record'].setOption(option);

                        //刷新告警数据
                        var option = that.charts['alert'].getOption();
                        var curdatas = [
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0)
                        ];
                        option.series[0].data = curdatas;

                        that.charts['alert'].setOption(option);

                    }, refreshtime);
                },
                /**
                 * 加载数据
                 */
                loadData: function() {
                    var that = this;
                    //加载内存数据
                    api.loadMemoryData({}, function(res) {
                        that.initMemoryChart(res);
                        that.initMemoryRecordChart(res);
                    });
                    //加载告警数据
                    api.loadAlertData({}, function(res) {
                        that.initMemoryAlertChart(res);
                    });
```

```
        },
        /**
         * 初始化 cpu 告警图表
         * @param {Object} res
         */
        initMemoryAlertChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('alertchart'));
            var colorArray = [
                '#1ace4a', //绿
                '#4bf3ff', //蓝
                '#b250ff', //粉
                '#ffa800' //黄
            ];
            var values = [];
            var leavels = [];
            res.data.leavel.forEach(function(item,index){
                leavels.push(item.label);
                var count = 0;
                for(var i= 0;i<res.data.values.length;i++){
                    if(res.data.values[i].from == 'memory' && res.data.values[i].type ==
item.value){
                        count += res.data.values[i].value
                    }
                }
                values.push(count);
            });
            var option = utils.createChartBaseOption('内存告警数据',
'50px',null,null,null,leavels);
            option.xAxis.boundaryGap = true;
            option.tooltip= {
                show: true,
                formatter: "{b}:{c}"
            };
            option.series= [{
                name: '告警数量',
                type: 'bar',
                label: {
                    normal: {
                        show: true,
                        position: 'top',
                        formatter: '{c}',
                        textStyle: {
                            color: 'white' //color of value
                        }
```

```
                        }
                    },
                    itemStyle: {
                        normal: {
                            show: true,
                            color: function(params) {
                                let num = colorArray.length;
                                return colorArray[params.dataIndex % num]
                            },
                            barBorderRadius: 20,
                            borderWidth: 0,
                            borderColor: '#333',
                        }
                    },
                    barWidth:'10%',
                    barGap: '0%',
                    barCategoryGap: '50%',
                    data: values
                }];
                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['alert'] = myChart;
            },
            /**
             * 初始化 cpu 图表
             * @param {Object} res
             */
            initMemoryRecordChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('recordchart'));

                var times = res.data.records.times;
                var option = utils.createChartBaseOption('内存使用情况',
'50px',null,null,null,times);
                option.series= {
                    name: '内存使用量',
                    symbolSize: 5,
                    type: "line",
                    smooth: true,
                    stack:'数量',
                    data: res.data.records.values,
                    itemStyle: {
                        normal: {
                            borderWidth: 1,
                            color: '#f56c6c',
```

```
                        shadowColor: 'rgba(93,241,255 ,0.7)',
                        shadowBlur: 10,
                        //shadowOffsetY: 0
                    }
                },
                label: {
                    show: true,
                    position: 'top',
                    textStyle: {
                        color: '#fff',
                    }
                }
            };
            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['record'] = myChart;
        },
        /**
         * 初始化 cpu 图表
         * @param {Object} res
         */
        initMemoryChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('memorychart'));

            var value = res.data.curdata; //百分比
            var lineheight = 150;
            var option = {
                tooltip: {
                    trigger: 'none'
                },
                grid:utils.createChartGaid('60px','60px',null,null),
                title: utils.createChartTitle('当前内存使用情况'),
                yAxis: {
                    data: ["内存使用"],
                    axisTick: {
                        show: false
                    },
                    axisLine: {
                        show: false
                    },
                    axisLabel: {
                        show: false
                    }
                },
```

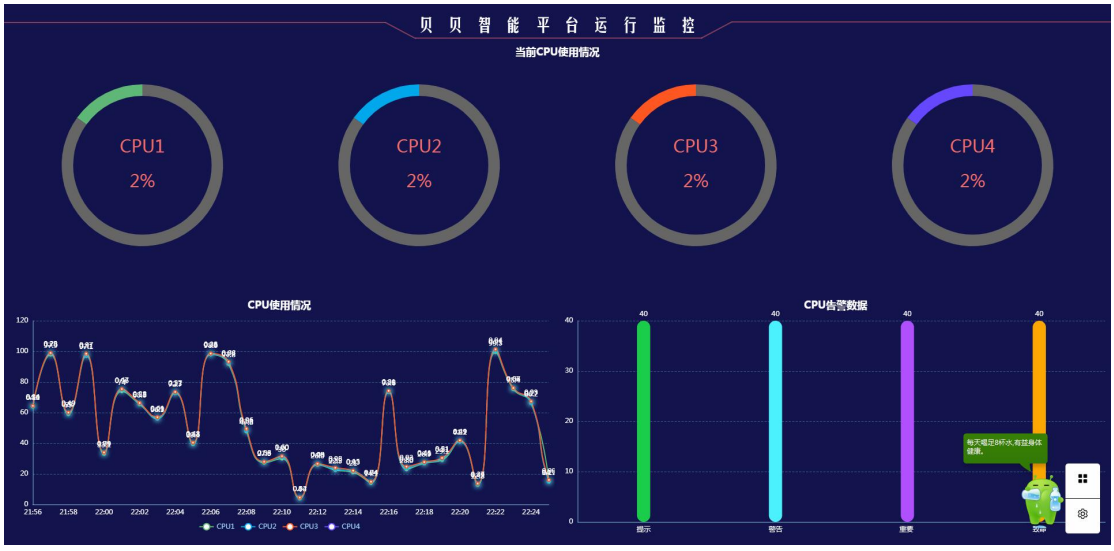
```
                xAxis: {
                    splitLine: {
                        show: false
                    },
                    axisTick: {
                        show: false
                    },
                    axisLine: {
                        show: false
                    },
                    axisLabel: {
                        show: false
                    }
                },
                series: [{
                    name: '最上层立体圆',
                    type: 'pictorialBar',
                    symbolSize: [45, lineheight],
                    symbolOffset: [20, 0],
                    z: 12,
                    itemStyle: {
                        normal: {
                            color: '#293CA0'
                        }
                    },
                    data: [{
                        value: 100,
                        symbolPosition: 'end'
                    }]
                }, {
                    name: '中间立体圆',
                    type: 'pictorialBar',
                    symbolSize: [45, lineheight],
                    symbolOffset: [20, 0],
                    z: 12,
                    itemStyle: {
                        normal: {
                            color: '#B687F9'
                        }
                    },
                    data: [{
                        value: value,
                        symbolPosition: 'end'
                    }]
```

```
        }, {
            name: '最底部立体圆',
            type: 'pictorialBar',
            symbolSize: [45, lineheight],
            symbolOffset: [-20, 0],
            z: 12,
            itemStyle: {
                normal: {
                    color: '#A052FE'
                }
            },
            data: [100 - value]
        }, {
            //底部立体柱
            stack: '1',
            type: 'bar',
            itemStyle: {
                normal: {
                    color: new echarts.graphic.LinearGradient(0, 0, 1, 0, [{
                        offset: 0,
                        color: '#CE7EFE'
                    }, {
                        offset: 1,
                        color: '#7125FF'
                    }])
                }
            },
            label: {
                show: true,
                position: "top",
                distance: 15,
                color: "#f56c6c",
                fontSize: 50,
                formatter: '{c}' + '%'
            },
            silent: true,
            barWidth: lineheight,
            barGap: '-100%', // Make series be overlap
            data: [value]
        }, {
            //上部立体柱
            stack: '1',
            type: 'bar',
            itemStyle: {
```

```
                                normal: {
                                    color: '#14257B',
                                    opacity: .7
                                }
                            },
                            silent: true,
                            barWidth: lineheight,
                            barGap: '-100%', // Make series be overlap
                            data: [100 - value]
                        }]
                    };

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['memory'] = myChart;
                }
            }
        }
```

8.3.5 CPU 监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Cpu.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="cpuchart">cpu 使用量
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="recordchart">历史使用情况
            </div>
        </el-col>
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="alertchart">告警信息
            </div>
        </el-col>
    </el-row>
</template>
```

4. 样式控制：

```
<style scoped="scoped">
    .title {
        height: v-bind(titleheight+"px");
        line-height: v-bind(titleheight+"px");
    }

    .chart_panel {
        height: v-bind(chartheight+"px");
    }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
```

```
import utils from '../public/utils.js'
import api from '../public/api.js'
import config from '../public/config.js'

import * as echarts from 'echarts'

export default {
    data() {
        return {
            chartheight: 100,
            titleheight: 60,
            timer: null,
            charts: []
        }
    },
    mounted: function() {
        this.chartheight = ($(window).height() - this.titleheight) / 2;
        this.loadData();

        //开始定时刷新报表数据
        this.startRefreshChart();

        var that = this;
        $(window).resize(function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            for (var key in that.charts) {
                that.charts[key].resize();
            }
        });
    },
    unmounted: function() {
        if (this.timer) {
            clearInterval(this.timer);
        }
    },
    methods: {
        /**
         * 定时刷新报表数据
         */
        startRefreshChart: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
            var that = this;
```

```
//获取刷新周期，TODO 配置变动时，此处需自动更新
var refreshtime = 60 * 1000;
config.getConfig().forEach(function(item, index) {
    if (item.key == 'refreshtime') {
        refreshtime = item.value;
    }
});

this.timer = setInterval(function() {
    //刷新 cpu 数据
    var option = that.charts['cpu'].getOption();

    var curdatas = [{
        "name": "CPU1",
        "value": (Math.random() * 100).toFixed(2)
    }, {
        "name": "CPU2",
        "value": (Math.random() * 100).toFixed(2)
    }, {
        "name": "CPU3",
        "value": (Math.random() * 100).toFixed(2)
    }, {
        "name": "CPU4",
        "value": (Math.random() * 100).toFixed(2)
    }];

    curdatas.forEach(function(item, index) {
        option.series[index].data[0].value = item.value;
        option.series[index].data[1].value = 100 - item.value;
    });
    that.charts['cpu'].setOption(option);

    //刷新历史数据
    var option = that.charts['record'].getOption();

    var times = [];
    var curdatas = [
        [],
        [],
        [],
        []
    ];
    var startData = new Date();
```

```
                        startData.setMinutes(startData.getMinutes() - 30);
                        for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                                1)) {
                            times.push(i.format('hh:mm'));
                            curdatas[0].push((Math.random(100) * 100).toFixed(2));
                            curdatas[1].push((Math.random(100) * 100).toFixed(2));
                            curdatas[2].push((Math.random(100) * 100).toFixed(2));
                            curdatas[3].push((Math.random(100) * 100).toFixed(2));
                        }
                        curdatas.forEach(function(item, index) {
                            option.series[index].data = item;
                        });
                        that.charts['record'].setOption(option);

                        //刷新告警数据
                        var option = that.charts['alert'].getOption();
                        var curdatas = [
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0)
                        ];
                        option.series[0].data = curdatas;

                        that.charts['alert'].setOption(option);

                    }, refreshtime);
                },
                /**
                 * 加载数据
                 */
                loadData: function() {
                    var that = this;
                    //加载 cpu 数据
                    api.loadCpuData({}, function(res) {
                        that.initCpuChart(res);
                        that.initCpuRecordChart(res);
                    });
                    //加载告警数据
                    api.loadAlertData({}, function(res) {
                        that.initCpuAlertChart(res);
                    });
                },
```

```
                /**
                 * 初始化 cpu 告警图表
                 * @param {Object} res
                 */
                initCpuAlertChart: function(res) {
                    var myChart = this.$echarts.init(document.getElementById('alertchart'));
                    var colorArray = [
                        '#1ace4a', //绿
                        '#4bf3ff', //蓝
                        '#b250ff', //粉
                        '#ffa800' //黄
                    ];
                    var values = [];
                    var leavels = [];
                    res.data.leavel.forEach(function(item,index){
                        leavels.push(item.label);
                        var count = 0;
                        for(var i= 0;i<res.data.values.length;i++){
                            if(res.data.values[i].from == 'cpu' && res.data.values[i].type == item.value){
                                count += res.data.values[i].value
                            }
                        }
                        values.push(count);
                    });
                    var option = utils.createChartBaseOption('CPU 告警数据', null,null,null,null,leavels);
                    option.xAxis.boundaryGap = true;
                    option.tooltip= {
                        show: true,
                        formatter: "{b}:{c}"
                    };
                    option.series= [{
                        name: '告警数量',
                        type: 'bar',
                        label: {
                            normal: {
                                show: true,
                                position: 'top',
                                formatter: '{c}',
                                textStyle: {
                                    color: 'white' //color of value
                                }
                            }
```

```
                },
                itemStyle: {
                    normal: {
                        show: true,
                        color: function(params) {
                            let num = colorArray.length;
                            return colorArray[params.dataIndex % num]
                        },
                        barBorderRadius: 20,
                        borderWidth: 0,
                        borderColor: '#333',
                    }
                },
                barWidth:'10%',
                barGap: '0%',
                barCategoryGap: '50%',
                data: values
            }];
            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['alert'] = myChart;
        },
        /**
         * 初始化 cpu 图表
         * @param {Object} res
         */
        initCpuRecordChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('recordchart'));

            var times = res.data.records.times;
            var legends = [];
            var series = [];
            var colors = ['#5FB878', '#01AAED', '#FF5722', '#6648FF'];
            res.data.cpus.forEach(function(item, index) {
                legends.push({
                    name: item
                });
                series.push({
                    name: item,
                    symbolSize: 5,
                    type: "line",
                    smooth: true,
                    stack:'数量',
                    data: res.data.records[item],
```

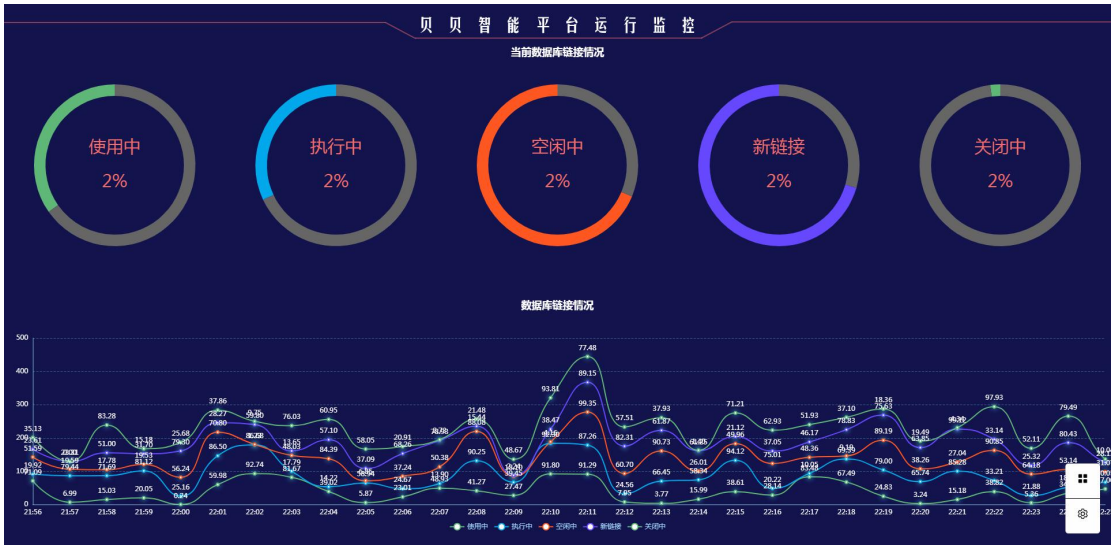
```
                        itemStyle: {
                            normal: {
                                borderWidth: 1,
                                color: colors[index % colors.length],
                                shadowColor: 'rgba(93,241,255 ,0.7)',
                                shadowBlur: 10,
                                //shadowOffsetY: 0
                            }
                        },
                        label: {
                            show: true,
                            position: 'top',
                            textStyle: {
                                color: '#fff',
                            }
                        }
                    });
                });

                var option = utils.createChartBaseOption('CPU 使 用 情 况 ', '50px',null,null,'70px',times);

                option.legend =  {
                    data: legends,
                    x: "center",
                    bottom: "20px",
                    textStyle: {
                        color: "#A1D5FF",
                        fontSize: 12
                    }
                };
                option.series=series;
                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['record'] = myChart;
            },
            /**
             * 初始化 cpu 图表
             * @param {Object} res
             */
            initCpuChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('cpuchart'));

                var dataStyle = {
```

```
            normal: {
                label: {
                    show: false
                },
                labelLine: {
                    show: false
                },
                shadowBlur: 0,
                shadowColor: '#203665'
            }
        };
        var colors = ['#5FB878', '#01AAED', '#FF5722', '#6648FF'];
        var series = [];
        var left = 100 / (res.data.curdatas.length);
        res.data.curdatas.forEach(function(item, index) {
            series.push({
                name: item.name,
                type: 'pie',
                clockWise: false,
                radius: [140, 120],
                itemStyle: dataStyle,
                hoverAnimation: false,
                center: [(left * (index + 1) - left / 2) + '%', '50%'],
                data: [{
                    value: item.value,
                    label: {
                        normal: {
                            rich: {
                                a: {
                                    color: colors[index % colors.length],
                                    align: 'center',
                                    fontSize: 30,
                                    fontWeight: "bold"
                                },
                                b: {
                                    color: '#f56c6c',
                                    align: 'center',
                                    fontSize: 30
                                }
                            },
                            formatter: function(params) {
                                return params.seriesName + "\n\n{b|2%}";
                            },
                            position: 'center',
```

```
                                        show: true,
                                        textStyle: {
                                            fontSize: '30',
                                            fontWeight: 'normal',
                                            color: '#f56c6c'
                                        }
                                    }
                                },
                                itemStyle: {
                                    normal: {
                                        color: colors[index % colors.length],
                                        shadowColor: colors[index % colors.length],
                                        shadowBlur: 0
                                    }
                                }
                            }, {
                                value: 100 - item.value,
                                name: '未使用',
                                itemStyle: {
                                    normal: {
                                        color: '#666'
                                    },
                                    emphasis: {
                                        color: '#666'
                                    }
                                }
                            }]
                        });
                    });

                    var option = {
                        title: utils.createChartTitle('当前 CPU 使用情况'),
                        tooltip: {
                            formatter: '{a} <br/>{b} : {c}%'
                        },
                        series: series
                    };

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['cpu'] = myChart;
                }
            }
        }
```

8.3.6 数据库监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Database.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="databasehart">
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="recordchart">
```

```
            </div>
        </el-col>
    </el-row>
</template>
```

4. 样式实现：

```
<style scoped="scoped">
    .title {
        height: v-bind(titleheight+"px");
        line-height: v-bind(titleheight+"px");
    }

    .chart_panel {
        height: v-bind(chartheight+"px");
    }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
    import utils from '../public/utils.js'
    import api from '../public/api.js'
    import config from '../public/config.js'

    import * as echarts from 'echarts'

    export default {
        data() {
            return {
                chartheight: 100,
                titleheight: 60,
                timer: null,
                charts: []
            }
        },
        mounted: function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            this.loadData();

            //开始定时刷新报表数据
            this.startRefreshChart();
```

```
        var that = this;
        $(window).resize(function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            for (var key in that.charts) {
                that.charts[key].resize();
            }
        });
    },
    unmounted: function() {
        if (this.timer) {
            clearInterval(this.timer);
        }
    },
    methods: {
        /**
         * 定时刷新报表数据
         */
        startRefreshChart: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
            var that = this;

            //获取刷新周期，TODO 配置变动时，此处需自动更新
            var refreshtime = 60 * 1000;
            config.getConfig().forEach(function(item, index) {
                if (item.key == 'refreshtime') {
                    refreshtime = item.value;
                }
            });

            this.timer = setInterval(function() {
                //刷新 cpu 数据
                var option = that.charts['database'].getOption();

                var curdatas = [{
                    "name": "使用中",
                    "value": (Math.random() * 100).toFixed(2)
                }, {
                    "name": "执行中",
                    "value": (Math.random() * 100).toFixed(2)
                }, {
                    "name": "空闲中",
```

```
                        "value": (Math.random() * 100).toFixed(2)
                    }, {
                        "name": "新链接",
                        "value": (Math.random() * 100).toFixed(2)
                    }, {
                        "name": "关闭中",
                        "value": (Math.random() * 100).toFixed(2)
                    }];

                    curdatas.forEach(function(item, index) {
                        option.series[index].data[0].value = item.value;
                        option.series[index].data[1].value = 100 - item.value;
                    });
                    that.charts['database'].setOption(option);

                    //刷新历史数据
                    var option = that.charts['record'].getOption();

                    var times = [];
                    var curdatas = [
                        [],
                        [],
                        [],
                        [],
                        []
                    ];
                    var startData = new Date();
                    startData.setMinutes(startData.getMinutes() - 30);
                    for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                            1)) {
                        times.push(i.format('hh:mm'));
                        curdatas[0].push((Math.random(100) * 100).toFixed(2));
                        curdatas[1].push((Math.random(100) * 100).toFixed(2));
                        curdatas[2].push((Math.random(100) * 100).toFixed(2));
                        curdatas[3].push((Math.random(100) * 100).toFixed(2));
                        curdatas[4].push((Math.random(100) * 100).toFixed(2));
                    }
                    curdatas.forEach(function(item, index) {
                        option.series[index].data = item;
                    });
                    that.charts['record'].setOption(option);

                }, refreshtime);
```

```
},
/**
 * 加载数据
 */
loadData: function() {
    var that = this;
    //加载内存数据
    api.loadDatabaseData({}, function(res) {
        that.initDatabaseChart(res);
        that.initRecordChart(res);
    });
},
/**
 * 初始化 cpu 图表
 * @param {Object} res
 */
initRecordChart: function(res) {
    var myChart = this.$echarts.init(document.getElementById('recordchart'));

    var times = res.data.records.times;
    var legends = [];
    var series = [];
    var colors = ['#5FB878', '#01AAED', '#FF5722', '#6648FF'];
    res.data.types.forEach(function(item, index) {
        legends.push({
            name: item.name
        });
        series.push({
            name: item.name,
            symbolSize: 5,
            type: "line",
            smooth: true,
            stack:'数量',
            data: res.data.records["_"+item.value],
            itemStyle: {
                normal: {
                    borderWidth: 1,
                    color: colors[index % colors.length],
                    shadowColor: 'rgba(93,241,255 ,0.7)',
                    shadowBlur: 10,
                }
            },
            label: {
                show: true,
```

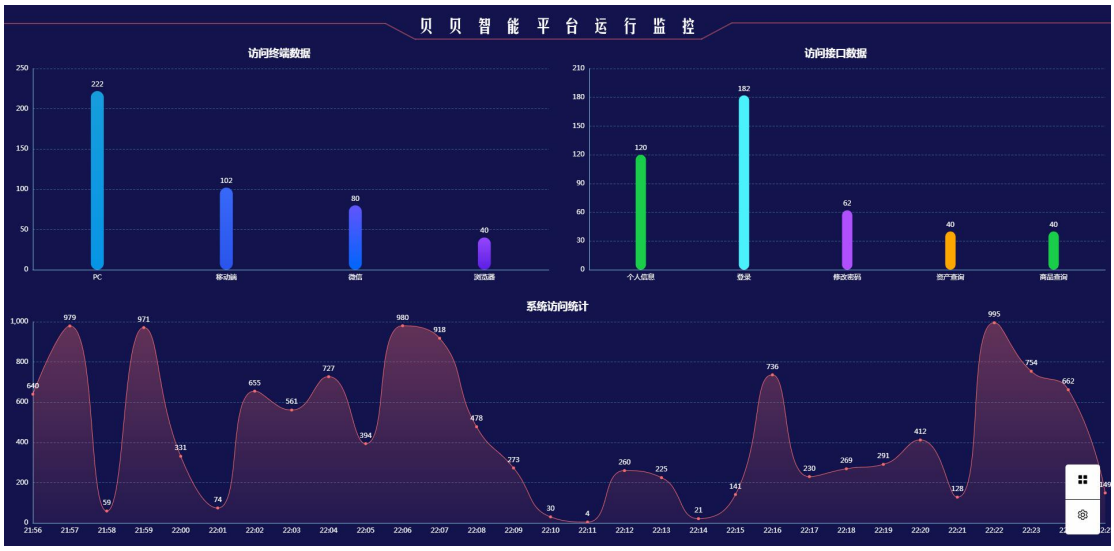
```
                                position: 'top',
                                textStyle: {
                                    color: '#fff',
                                }
                            }
                        });
                    });

                    var option = utils.createChartBaseOption('数据库链接情况',
'50px',null,'80px','70px',times);
                    option.legend =  {
                        data: legends,
                        x: "center",
                        bottom: "20px",
                        textStyle: {
                            color: "#A1D5FF",
                            fontSize: 12
                        },
                    };
                    option.series = series;

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['record'] = myChart;
                },
                /**
                 * 初始化 cpu 图表
                 * @param {Object} res
                 */
                initDatabaseChart: function(res) {
                    var myChart = this.$echarts.init(document.getElementById('databasehart'));

                    var dataStyle = {
                        normal: {
                            label: {
                                show: false
                            },
                            labelLine: {
                                show: false
                            },
                            shadowBlur: 0,
                            shadowColor: '#203665'
                        }
                    };
```

```
var colors = ['#5FB878', '#01AAED', '#FF5722', '#6648FF'];
var series = [];
var left = 100 / (res.data.types.length);
res.data.types.forEach(function(item, index) {
    var value = 0;
    for (var i = 0; i < res.data.curdata.length; i++) {
        if(res.data.curdata[i].type == item.value){
            value += res.data.curdata[i].value;
        }
    }
    series.push({
        name: item.name,
        type: 'pie',
        clockWise: false,
        radius: [140, 120],
        itemStyle: dataStyle,
        hoverAnimation: false,
        center: [(left * (index + 1) - left / 2) + '%', '50%'],
        data: [{
            value: value,
            label: {
                normal: {
                    rich: {
                        a: {
                            color: colors[index % colors.length],
                            align: 'center',
                            fontSize: 30,
                            fontWeight: "bold"
                        },
                        b: {
                            color: '#f56c6c',
                            align: 'center',
                            fontSize: 30
                        }
                    },
                    formatter: function(params) {
                        return params.seriesName + "\n\n{b|2%}";
                    },
                    position: 'center',
                    show: true,
                    textStyle: {
                        fontSize: '30',
                        fontWeight: 'normal',
                        color: '#f56c6c'
```

```
                                        }
                                    }
                                },
                                itemStyle: {
                                    normal: {
                                        color: colors[index % colors.length],
                                        shadowColor: colors[index % colors.length],
                                        shadowBlur: 0
                                    }
                                }
                            }, {
                                value: 100 - value,
                                name: '全部',
                                itemStyle: {
                                    normal: {
                                        color: '#666'
                                    },
                                    emphasis: {
                                        color: '#666'
                                    }
                                }
                            }]
                        });
                    });

                    var option = {
                        title:utils.createChartTitle('当前数据库链接情况'),
                        tooltip: {
                            formatter: '{a} <br/>{b} : {c}%'
                        },
                        series: series
                    };

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['database'] = myChart;
                }
            }
        }
```

8.3.7 访问量监控界面实现

1. 效果预览：



2. 文件路径：

./pages/View.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="clientchart">终端分布
            </div>
        </el-col>
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="urlchart">接口分布
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
```

```
            <el-col >
                <div class="grid-content bg-purple chart_panel" id="recordchart">请求量
                </div>
            </el-col>
        </el-row>
</template>
```

4. 样式实现：

```
<style scoped="scoped">
        .title {
            height: v-bind(titleheight+"px");
            line-height: v-bind(titleheight+"px");
        }

        .chart_panel {
            height: v-bind(chartheight+"px");
        }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
        import utils from '../public/utils.js'
        import api from '../public/api.js'
        import config from '../public/config.js'

        import * as echarts from 'echarts'

        export default {
            data() {
                return {
                    chartheight: 100,
                    titleheight: 60,
                    timer: null,
                    charts: []
                }
            },
            mounted: function() {
                this.chartheight = ($(window).height() - this.titleheight) / 2;
                this.loadData();
```

```
            //开始定时刷新报表数据
            this.startRefreshChart();

            var that = this;
            $(window).resize(function() {
                this.chartheight = ($(window).height() - this.titleheight) / 2;
                for (var key in that.charts) {
                    that.charts[key].resize();
                }
            });
        },
        unmounted: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
        },
        methods: {
            /**
             * 定时刷新报表数据
             */
            startRefreshChart: function() {
                if (this.timer) {
                    clearInterval(this.timer);
                }
                var that = this;

                //获取刷新周期，TODO 配置变动时，此处需自动更新
                var refreshtime = 60 * 1000;
                config.getConfig().forEach(function(item, index) {
                    if (item.key == 'refreshtime') {
                        refreshtime = item.value;
                    }
                });

                this.timer = setInterval(function() {
                    //更新访问情况
                    var option = that.charts['record'].getOption();
                    var startData = new Date();
                    var times = [];var values = [];
                    startData.setMinutes(startData.getMinutes() - 30);
                    for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() + 1)) {
                        times.push(i.format('hh:mm'));
                        values.push((Math.random(1000) * 1000).toFixed(0));
```

```
                }
                option.xAxis[0].data =times;
                option.series[0].data =values;
                that.charts['record'].setOption(option);

                //更新客户端情况
                var option = that.charts['client'].getOption();
                var curdatas = [
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0)
                ];
                option.series[0].data = curdatas;

                that.charts['client'].setOption(option);

                //更新接口情况
                var option = that.charts['url'].getOption();
                var curdatas = [
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0),
                    (Math.random(100) * 100).toFixed(0)
                ];
                option.series[0].data = curdatas;

                that.charts['url'].setOption(option);

            }, refreshtime);
        },
        /**
         * 加载数据
         */
        loadData: function() {
            var that = this;
            //加载内存数据
            api.loadViewData({}, function(res) {
                that.initViewRecordChart(res);
                that.initViewClientChart(res);
                that.initViewUrlChart(res);
            });
        },
```

```
            /**
             * 初始化 cpu 告警图表
             * @param {Object} res
             */
            initViewRecordChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('recordchart'));

                var times = res.data.times;
                var values = res.data.values;

                var option = utils.createChartBaseOption('系统访问统计', '50px',null,null,null,times);
                option.tooltip= {
                    trigger: 'axis'
                },
                option.series= {
                    name: '访问量',
                    type: 'line',
                    color: '#0092f6',
                    smooth: true,
                    itemStyle: {
                        normal: {
                            color: '#f56c6c',
                            lineStyle: {
                                color: '#f56c6c',
                                width: 1,
                            },
                            areaStyle: {
                                color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [{
                                    offset: 0,
                                    color: 'rgba(245,108,108, 0.5)'
                                }, {
                                    offset: 1,
                                    color: 'rgba(245,108,108, 0.1)'
                                }], false)
                            }
                        }
                    },
                    label: {
                        show: true,
                        position: 'top',
                        textStyle: {
                            color: '#fff',
                        }
```

```
                },
                symbol: 'circle',
                symbolSize: 5,
                data: values
            };

            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['record'] = myChart;
        },
        /**
         * 初始化 cpu 图表
         * @param {Object} res
         */
        initViewClientChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('clientchart'));

            var colorArray = [
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(23, 158, 221, 1)' },
                    { offset: 1, color: 'rgba(8, 150, 231, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(57, 105, 250, 1)' },
                    { offset: 1, color: 'rgba(41, 85, 237, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(98, 85, 255, 1)' },
                    { offset: 1, color: 'rgba(0, 102, 255, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(150, 71, 254, 1)' },
                    { offset: 1, color: 'rgba(92, 31, 228, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(205, 100, 250, 1)' },
                    { offset: 1, color: 'rgba(149, 70, 254, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(242, 10, 247, 1)' },
                    { offset: 1, color: 'rgba(171, 28, 221, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(247, 10, 144, 1)' },
```

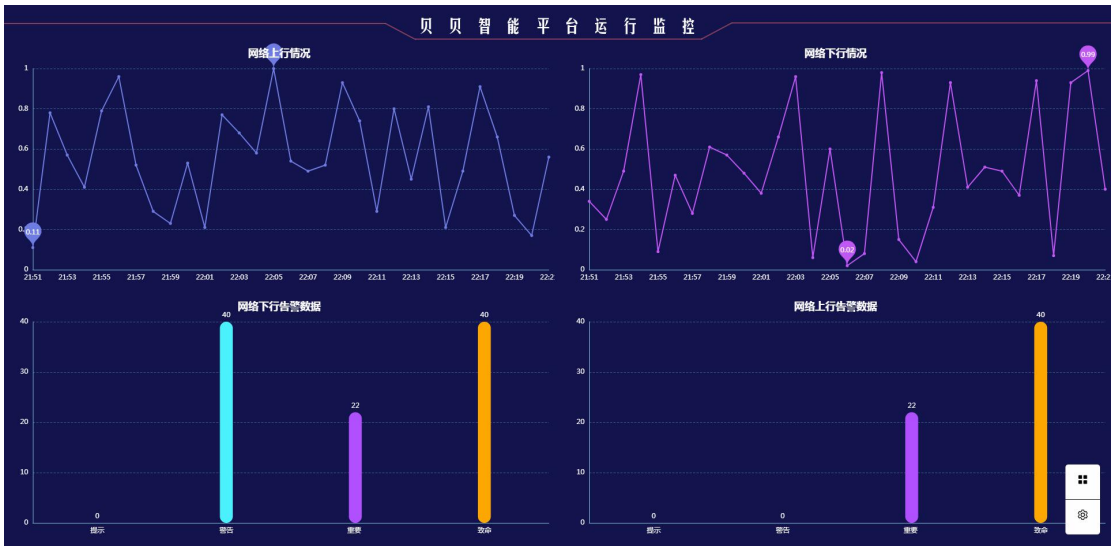
```
                        { offset: 1, color: 'rgba(176, 1, 180, 1)' },
                    ])
                ];
                var values = [];
                var clients = [];
                res.data.clients.forEach(function(item, index) {
                    clients.push(item.label);
                    var count = 0;
                    for (var i = 0; i < res.data.record.length; i++) {
                        if (res.data.record[i].clients == item.value) {
                            count += res.data.record[i].value
                        }
                    }
                    values.push(count);
                });
                var option = utils.createChartBaseOption('访问终端数据',
'50px',null,null,null,clients);
                option.xAxis.boundaryGap = true;
                option.tooltip= {
                    show: true,
                    formatter: "{b}:{c}"
                };
                option.series= [{
                    name: '访问数量',
                    type: 'bar',
                    label: {
                        normal: {
                            show: true,
                            position: 'top',
                            formatter: '{c}',
                            textStyle: {
                                color: 'white' //color of value
                            }
                        }
                    },
                    itemStyle: {
                        normal: {
                            show: true,
                            color: function(params) {
                                let num = colorArray.length;
                                return colorArray[params.dataIndex % num]
                            },
                            barBorderRadius: 20,
                            borderWidth: 0,
```

```
                                borderColor: '#333',
                            }
                        },
                        barWidth: '10%',
                        barGap: '0%',
                        barCategoryGap: '50%',
                        data: values
                    }];

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['client'] = myChart;
                },
                /**
                 * 初始化 cpu 图表
                 * @param {Object} res
                 */
                initViewUrlChart: function(res) {
                    var myChart = this.$echarts.init(document.getElementById('urlchart'));

                    var colorArray = [
                        '#1ace4a', //绿
                        '#4bf3ff', //蓝
                        '#b250ff', //粉
                        '#ffa800' //黄
                    ];
                    var values = [];
                    var urls = [];
                    res.data.urls.forEach(function(item, index) {
                        urls.push(item.label);
                        var count = 0;
                        for (var i = 0; i < res.data.record.length; i++) {
                            if (res.data.record[i].url == item.value) {
                                count += res.data.record[i].value
                            }
                        }
                        values.push(count);
                    });
                    var option = utils.createChartBaseOption('访问接口数据',
'50px',null,null,null,urls);
                    option.xAxis.boundaryGap = true;
                    option.tooltip= {
                        show: true,
                        formatter: "{b}:{c}"
```

```
                    };
                    option.series = [{
                        name: '访问数量',
                        type: 'bar',
                        label: {
                            normal: {
                                show: true,
                                position: 'top',
                                formatter: '{c}',
                                textStyle: {
                                    color: 'white' //color of value
                                }
                            }
                        },
                        itemStyle: {
                            normal: {
                                show: true,
                                color: function(params) {
                                    let num = colorArray.length;
                                    return colorArray[params.dataIndex % num]
                                },
                                barBorderRadius: 20,
                                borderWidth: 0,
                                borderColor: '#333',
                            }
                        },
                        barWidth: '10%',
                        barGap: '0%',
                        barCategoryGap: '50%',
                        data: values
                    }];

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['url'] = myChart;
                }
            }
        }
```

8.3.8 网络监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Net.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="netinchart">网络使用量
            </div>
        </el-col>
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="netoutchart">网络使用量
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
```

```
            <el-col :span="12">
                <div class="grid-content bg-purple chart_panel" id="alertinchart">告警信息
                </div>
            </el-col>
            <el-col :span="12">
                <div class="grid-content bg-purple chart_panel" id="alertoutchart">告警信息
                </div>
            </el-col>
        </el-row>
</template>
```

4. 样式实现：

```
<style scoped="scoped">
        .title {
            height: v-bind(titleheight+"px");
            line-height: v-bind(titleheight+"px");
        }

        .chart_panel {
            height: v-bind(chartheight+"px");
        }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
        import utils from '../public/utils.js'
        import api from '../public/api.js'
        import config from '../public/config.js'

        import * as echarts from 'echarts'

        export default {
            data() {
                return {
                    chartheight: 100,
                    titleheight: 60,
                    timer: null,
                    charts: []
                }
            },
```

```
mounted: function() {
    this.chartheight = ($(window).height() - this.titleheight) / 2;
    this.loadData();

    //开始定时刷新报表数据
    this.startRefreshChart();

    var that = this;
    $(window).resize(function() {
        this.chartheight = ($(window).height() - this.titleheight) / 2;
        for (var key in that.charts) {
            that.charts[key].resize();
        }
    });
},
unmounted: function() {
    if (this.timer) {
        clearInterval(this.timer);
    }
},
methods: {
    /**
     * 定时刷新报表数据
     */
    startRefreshChart: function() {
        if (this.timer) {
            clearInterval(this.timer);
        }
        var that = this;

        //获取刷新周期，TODO 配置变动时，此处需自动更新
        var refreshtime = 60 * 1000;
        config.getConfig().forEach(function(item, index) {
            if (item.key == 'refreshtime') {
                refreshtime = item.value;
            }
        });

        this.timer = setInterval(function() {
            //刷新 cpu 数据
            var option = that.charts['netin'].getOption();
            var times = [];
            var curdatas = [];
            var startData = new Date();
```

```
                        startData.setMinutes(startData.getMinutes() - 30);
                        for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                                1)) {
                            times.push(i.format('hh:mm'));
                            curdatas.push((Math.random(100) * 100).toFixed(2));
                        }
                        option.series[0].data = curdatas;
                        that.charts['netin'].setOption(option);

                        var option = that.charts['netout'].getOption();
                        var times = [];
                        var curdatas = [];
                        var startData = new Date();
                        startData.setMinutes(startData.getMinutes() - 30);
                        for (var i = startData; i.getTime() < new Date().getTime();
i.setMinutes(i.getMinutes() +
                                1)) {
                            times.push(i.format('hh:mm'));
                            curdatas.push((Math.random(100) * 100).toFixed(2));
                        }
                        option.series[0].data = curdatas;
                        that.charts['netout'].setOption(option);


                        //刷新告警数据
                        var option = that.charts['alertin'].getOption();
                        var curdatas = [
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0)
                        ];
                        option.series[0].data = curdatas;
                        that.charts['alertin'].setOption(option);

                        var option = that.charts['alertout'].getOption();
                        var curdatas = [
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0),
                            (Math.random(100) * 100).toFixed(0)
                        ];
                        option.series[0].data = curdatas;
```

```
                    that.charts['alertout'].setOption(option);

                }, refreshtime);
            },
            /**
             * 加载数据
             */
            loadData: function() {
                var that = this;
                //加载网络数据
                api.loadNetData({}, function(res) {
                    that.initNetInChart(res);
                    that.initNetOutChart(res);
                });
                //加载告警数据
                api.loadAlertData({}, function(res) {
                    that.initNetInAlertChart(res);
                    that.initNetOutAlertChart(res);
                });
            },
            /**
             * 初始化网络告警图表
             * @param {Object} res
             */
            initNetOutAlertChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('alertinchart'));
                var colorArray = [
                    '#1ace4a', //绿
                    '#4bf3ff', //蓝
                    '#b250ff', //粉
                    '#ffa800' //黄
                ];
                var values = [];
                var leavels = [];
                res.data.leavel.forEach(function(item,index){
                    leavels.push(item.label);
                    var count = 0;
                    for(var i= 0;i<res.data.values.length;i++){
                        if(res.data.values[i].from == 'netout' && res.data.values[i].type ==
item.value){
                            count += res.data.values[i].value
                        }
                    }
                    values.push(count);
```

```
                });
                var option = utils.createChartBaseOption('网络下行告警数据', '50px',null,null,null,leavels);
                option.xAxis.boundaryGap = true;
                option.tooltip= {
                    show: true,
                    formatter: "{b}:{c}"
                },
                option.series= [{
                    name: '告警数量',
                    type: 'bar',
                    label: {
                        normal: {
                            show: true,
                            position: 'top',
                            formatter: '{c}',
                            textStyle: {
                                color: 'white' //color of value
                            }
                        }
                    },
                    itemStyle: {
                        normal: {
                            show: true,
                            color: function(params) {
                                let num = colorArray.length;
                                return colorArray[params.dataIndex % num]
                            },
                            barBorderRadius: 20,
                            borderWidth: 0,
                            borderColor: '#333',
                        }
                    },
                    barWidth:'10%',
                    barGap: '0%',
                    barCategoryGap: '50%',
                    data: values
                }];
                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['alertin'] = myChart;
            },
            /**
             * 初始化网络告警图表
```

```
                * @param {Object} res
                */
               initNetInAlertChart: function(res) {
                    var myChart = this.$echarts.init(document.getElementById('alertoutchart'));
                    var colorArray = [
                         '#1ace4a', //绿
                         '#4bf3ff', //蓝
                         '#b250ff', //粉
                         '#ffa800' //黄
                    ];
                    var values = [];
                    var leavels = [];
                    res.data.leavel.forEach(function(item,index){
                         leavels.push(item.label);
                         var count = 0;
                         for(var i= 0;i<res.data.values.length;i++){
                              if(res.data.values[i].from == 'netin' && res.data.values[i].type ==
item.value){
                                   count += res.data.values[i].value
                              }
                         }
                         values.push(count);
                    });
                    var option = utils.createChartBaseOption('网络上行告警数据',
'50px',null,null,null,leavels);
                    option.xAxis.boundaryGap = true;
                    option.tooltip= {
                         show: true,
                         formatter: "{b}:{c}"
                    };
                    option.series= [{
                         name: '告警数量',
                         type: 'bar',
                         label: {
                              normal: {
                                   show: true,
                                   position: 'top',
                                   formatter: '{c}',
                                   textStyle: {
                                        color: 'white' //color of value
                                   }
                              }
                         },
                         itemStyle: {
```

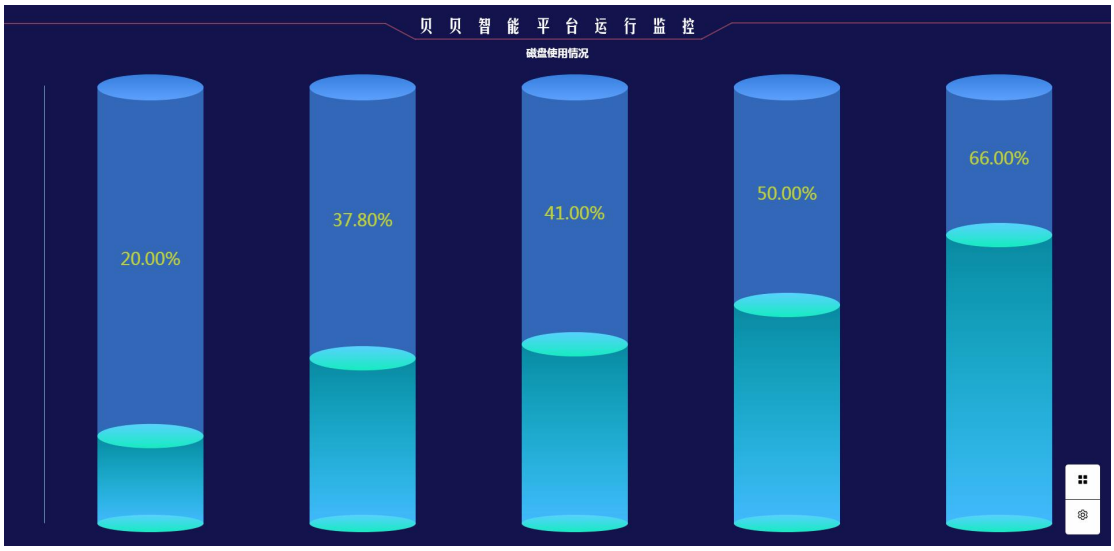
```
                                normal: {
                                    show: true,
                                    color: function(params) {
                                        let num = colorArray.length;
                                        return colorArray[params.dataIndex % num]
                                    },
                                    barBorderRadius: 20,
                                    borderWidth: 0,
                                    borderColor: '#333',
                                }
                            },
                            barWidth:'10%',
                            barGap: '0%',
                            barCategoryGap: '50%',
                            data: values
                        }];
                        // 使用刚指定的配置项和数据显示图表。
                        myChart.setOption(option);
                        this.charts['alertout'] = myChart;
                    },
                    /**
                     * 初始化网络图表
                     * @param {Object} res
                     */
                    initNetOutChart: function(res) {
                        var myChart = this.$echarts.init(document.getElementById('netoutchart'));
                        var xData = res.data.times;
                        var values = res.data.outs;
                        var option = utils.createChartBaseOption('网络下行情况',
'50px',null,null,null,xData);
                        option.tooltip= {
                            trigger: "axis",
                            axisPointer: {
                                type: "shadow",
                                textStyle: {
                                    color: "#fff",
                                }
                            }
                        },
                        option.calculable= true,
                        option.series= [{
                            name: "下行",
                            type: "line",
                            symbolSize: 5,
```

```
                    symbol: 'circle',
                    itemStyle: {
                        color: "#c257F6",
                    },
                    markPoint: {
                        label: {
                            normal: {
                                textStyle: {
                                    color: '#fff'
                                }
                            }
                        },
                        data: [{
                            type: 'max',
                            name: '最大值',

                        }, {
                            type: 'min',
                            name: '最小值'
                        }]
                    },
                    data: values
                } ];
                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['netout'] = myChart;
            },
            /**
             * 初始化网络图表
             * @param {Object} res
             */
            initNetInChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('netinchart'));
                var xData = res.data.times;
                var values = res.data.ins;
                var option = utils.createChartBaseOption('网络上行情况',
'50px',null,null,null,xData);
                option.tooltip= {
                    trigger: "axis",
                    axisPointer: {
                        type: "shadow",
                        textStyle: {
                            color: "#fff",
                        }
```

```
                    }
                };
                option.calculable= true;
                option.series= [{
                    name: "上行",
                    type: "line",
                    symbolSize: 5,
                    symbol: 'circle',
                    itemStyle: {
                        color: "#6f7de3",
                    },
                    markPoint: {
                        label: {
                            normal: {
                                textStyle: {
                                    color: '#fff'
                                }
                            }
                        },
                        data: [{
                            type: 'max',
                            name: '最大值',

                        }, {
                            type: 'min',
                            name: '最小值'
                        }]
                    },
                    data: values
                } ];
                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['netin'] = myChart;
            }
        }
    }
```

8.3.9 磁盘监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Dist.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="distchart">网络使用量
            </div>
        </el-col>
    </el-row>
</template>
```

4. 样式实现：

```
<style scoped="scoped">
	.title {
		height: v-bind(titleheight+"px");
		line-height: v-bind(titleheight+"px");
	}

	.chart_panel {
		height: v-bind(chartheight+"px");
	}
</style>
```

5. JS 代码：

```
import $ from 'jquery'
	import utils from '../public/utils.js'
	import api from '../public/api.js'
	import config from '../public/config.js'

	import * as echarts from 'echarts'

	export default {
		data() {
			return {
				chartheight: 100,
				titleheight: 60,
				timer: null,
				charts: []
			}
		},
		mounted: function() {
			this.chartheight = ($(window).height() - this.titleheight);
			this.loadData();

			//开始定时刷新报表数据
			this.startRefreshChart();

			var that = this;
			$(window).resize(function() {
				this.chartheight = ($(window).height() - this.titleheight);
				for (var key in that.charts) {
```

```
                    that.charts[key].resize();
                }
            });
        },
        unmounted: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
        },
        methods: {
            /**
             * 定时刷新报表数据
             */
            startRefreshChart: function() {
                if (this.timer) {
                    clearInterval(this.timer);
                }
                var that = this;

                //获取刷新周期，TODO 配置变动时，此处需自动更新
                var refreshtime = 60 * 1000;
                config.getConfig().forEach(function(item, index) {
                    if (item.key == 'refreshtime') {
                        refreshtime = item.value;
                    }
                });

                this.timer = setInterval(function() {
                    //刷新 cpu 数据
                    var option = that.charts['dist'].getOption();

                    for (var i = 0; i < 5; i++) {
                        var value = (Math.random(100) * 100);
                        option.series[0].data[i].value = 100;
                        option.series[1].data[i].value = value.toFixed(2);
                        option.series[2].data[i].value =    100;
                        option.series[3].data[i].value =    value.toFixed(2);
                        option.series[4].data[i].value =    (100 - value).toFixed(2);
                    }

                    that.charts['dist'].setOption(option);

                }, refreshtime);
            },
```

```
/**
 * 加载数据
 */
loadData: function() {
	var that = this;
	//加载内存数据
	api.loadDistData({}, function(res) {
		that.initDistChart(res);
	});
},
/**
 * 初始化 cpu 图表
 * @param {Object} res
 */
initDistChart: function(res) {
	var myChart = this.$echarts.init(document.getElementById('distchart'));

	var categorys = [];
	var data1 = [];
	var data2 = [];
	var data3 = [];
	var data4 = [];
	var data5 = [];
	res.data.forEach(function(item, index) {
		categorys.push(item.name);
		var value = 0;
		for (var i = 0; i < res.data.length; i++) {
			if (res.data[i].name == item.name) {
				value = res.data[i].value
			}
		}
		data1.push({
			name: "",
			value: 100,
			symbolPosition: "end"
		});
		data2.push({
			value: value
		});
		data3.push({
			name: "",
			value: "100"
		});
		data4.push({
```

```
                        name: "",
                        value: value,
                        symbolPosition: "end"
                    });
                    data5.push({
                        name: "a",
                        value: 100 - value
                    });
                });
                var barwidth = '50%';
                var size = '65%';

                var option = utils.createChartBaseOption('磁盘使用情况',
'70px',null,'80px',null,categorys);
                option.xAxis.axisLine = false
                option.xAxis.axisLabel = false
                option.xAxis.boundaryGap = true;
                option.yAxis.splitLine = false;
                option.yAxis.axisLabel = false;
                option.series = [{ // 头
                        name: "",
                        type: "pictorialBar",
                        symbolSize: [size, 45],
                        symbolOffset: [0, -20],
                        z: 12,
                        itemStyle: {
                            normal: {
                                color: new echarts.graphic.LinearGradient(0, 0, 0, 1,
                                    [{
                                            offset: 0,
                                            color: "rgba(54,127,223,1)"
                                        },
                                        {
                                            offset: 1,
                                            color: "rgba(94,162,254,1)"
                                        }
                                    ],
                                    false
                                ),
                            }
                        },
                        data: data1
                    },
                    //底部立体柱
```

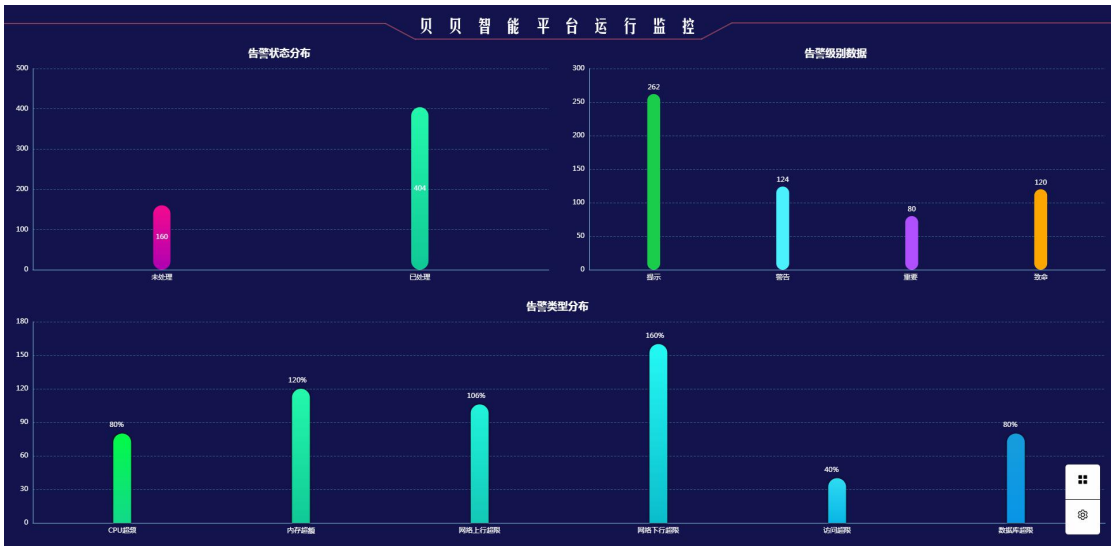
```
                {
                    name: "vvvv",
                    stack: '1',
                    type: 'bar',
                    silent: true,
                    barWidth: barwidth,
                    barGap: '-100%', // Make series be overlap
                    data: data2,
                    itemStyle: {
                        normal: {
                            color: {
                                x: 0,
                                y: 0,
                                x2: 0,
                                y2: 1,
                                type: "linear",
                                global: false,
                                colorStops: [{ //第一节下面
                                    offset: 0,
                                    color: "rgba(0,255,245,0.5)"
                                }, {
                                    offset: 1,
                                    color: "#43bafe"
                                }]
                            }
                        }
                    }
                },
                //三个最低下的圆片
                {
                    name: "",
                    type: "pictorialBar",
                    symbolSize: [size, 30],
                    symbolOffset: [0, 16],
                    z: 12,
                    itemStyle: {
                        normal: {
                            color: new echarts.graphic.LinearGradient(0, 0, 0, 1, [{
                                    offset: 0,
                                    color: "rgba(89,211,255,1)"
                                },
                                {
                                    offset: 1,
                                    color: "rgba(23,237,194,1)"
```

```
                        }
                    ])
                }
            },
            data: data3
        },
        // 中间圆片
        {
            name: "",
            type: "pictorialBar",
            symbolSize: [size, 42],
            symbolOffset: [0, -20],
            itemStyle: {
                normal: {
                    color: new echarts.graphic.LinearGradient(0, 0, 0, 1,
                        [{
                                offset: 0,
                                color: "rgba(89,211,255,1)"
                            },
                            {
                                offset: 1,
                                color: "rgba(23,237,194,1)"
                            }
                        ],
                        false
                    ),
                }
            },
            z: 12,
            data: data4
        },
        //上部立体柱
        {
            //上部立体柱
            stack: '1',
            type: 'bar',
            itemStyle: {
                normal: {
                    color: '#3E8BE6',
                    opacity: .7
                }
            },
            label: {
                show: true,
```

```
                                position: 'inside',
                                distance: 20,
                                color: "#FFFE00",
                                fontSize: 30,
                                formatter: function(item) {
                                    var a = 100
                                    return (a - item.value).toFixed(2) + '%'
                                }
                            },
                            silent: true,
                            barWidth: barwidth,
                            barGap: '-100%', // Make series be overlap
                            data: data5
                        }
                    ];

                    // 使用刚指定的配置项和数据显示图表。
                    myChart.setOption(option);
                    this.charts['dist'] = myChart;
                }
            }
        }
```

8.3.10 告警监控界面实现

1. 效果预览：



2. 文件路径：

./pages/Alert.vue

3. 界面布局：

```
<template>
    <div class="title">
        <!-- 贝贝智能平台运行监控 -->
        <img src="../assets/title_bg.png">
    </div>
    <el-row :gutter="10">
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="statuschart">状态分布
            </div>
        </el-col>
        <el-col :span="12">
            <div class="grid-content bg-purple chart_panel" id="leavelchart">级别分布
            </div>
        </el-col>
    </el-row>
    <el-row :gutter="10">
        <el-col>
            <div class="grid-content bg-purple chart_panel" id="typechart">类型分布
            </div>
        </el-col>
    </el-row>
</template>
```

4. 样式实现：

```
<style scoped="scoped">
    .title {
        height: v-bind(titleheight+"px");
        line-height: v-bind(titleheight+"px");
    }

    .chart_panel {
        height: v-bind(chartheight+"px");
    }
</style>
```

5. JS 代码：

```
import $ from 'jquery'
    import utils from '../public/utils.js'
    import api from '../public/api.js'
    import config from '../public/config.js'

    import * as echarts from 'echarts'

    export default {
        data() {
            return {
                chartheight: 100,
                titleheight: 60,
                timer: null,
                charts: []
            }
        },
        mounted: function() {
            this.chartheight = ($(window).height() - this.titleheight) / 2;
            this.loadData();

            //开始定时刷新报表数据
            this.startRefreshChart();

            var that = this;
            $(window).resize(function() {
                this.chartheight = ($(window).height() - this.titleheight) / 2;
                for (var key in that.charts) {
                    that.charts[key].resize();
                }
            });
        },
        unmounted: function() {
            if (this.timer) {
                clearInterval(this.timer);
            }
        },
        methods: {
            /**
             * 定时刷新报表数据
             */
            startRefreshChart: function() {
```

```
if (this.timer) {
    clearInterval(this.timer);
}
var that = this;

//获取刷新周期，TODO 配置变动时，此处需自动更新
var refreshtime = 60 * 1000;
config.getConfig().forEach(function(item, index) {
    if (item.key == 'refreshtime') {
        refreshtime = item.value;
    }
});

this.timer = setInterval(function() {
    var option = that.charts['type'].getOption();
    var curdatas = [
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0)
    ];
    option.series[0].data = curdatas;

    that.charts['type'].setOption(option);

    var option = that.charts['status'].getOption();
    var curdatas = [
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0)
    ];
    option.series[0].data = curdatas;

    that.charts['status'].setOption(option);

    var option = that.charts['leavel'].getOption();
    var curdatas = [
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0),
        (Math.random(100) * 100).toFixed(0)
    ];
    option.series[0].data = curdatas;
```

```
                that.charts['leavel'].setOption(option);

            }, refreshtime);
        },
        /**
         * 加载数据
         */
        loadData: function() {
            var that = this;
            //加载告警数据
            api.loadAlertData({}, function(res) {
                that.initStatusAlertChart(res);
                that.initLeabelRecordChart(res);
                that.initTypeChart(res);
            });
        },
        /**
         * 初始化告警类型图表
         * @param {Object} res
         */
        initStatusAlertChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('statuschart'));

            var colorList = [
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(247, 10, 144, 1)' },
                    { offset: 1, color: 'rgba(176, 1, 180, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(36, 250, 173, 1)' },
                    { offset: 1, color: 'rgba(16, 202, 151, 1)' },
                ])
            ];

            var status = [];
            var values = [];
            res.data.status.forEach(function(item,index){
                status.push(item.label);
                var value    = 0;
                for (var i = 0; i < res.data.values.length; i++) {
                    if(res.data.values[i].status == item.value){
                        value += res.data.values[i].value;
                    }
```

```
                    }
                    values.push(value);
                })
                var option = utils.createChartBaseOption('告警状态分布', '50px',null,null,null,status);
                option.xAxis.boundaryGap = true;
                option.tooltip= {
                    show: true,
                    formatter: "{b}:{c}"
                },
                option.series= [{
                    zlevel: 1,
                    name: '',
                    type: 'bar',
                    barWidth: '30px',
                    data: values,
                    align: 'center',
                    itemStyle: {
                        normal: {
                            color: function (params) {
                                return colorList[params.dataIndex];
                            },
                            barBorderRadius: 30
                        }
                    },
                    label: {
                        show: true,
                        color: '#fff'
                    }
                }];

                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['status'] = myChart;
            },
            /**
             * 初始化告警级别
             * @param {Object} res
             */
            initLeabelRecordChart: function(res) {
                var myChart = this.$echarts.init(document.getElementById('leavelchart'));

                var colorArray = [
                    '#1ace4a', //绿
```

```
                    '#4bf3ff', //蓝
                    '#b250ff', //粉
                    '#ffa800' //黄
                ];
                var values = [];
                var leavels = [];
                res.data.leavel.forEach(function(item, index) {
                    leavels.push(item.label);
                    var count = 0;
                    for (var i = 0; i < res.data.values.length; i++) {
                        if (res.data.values[i].leavel == item.value) {
                            count += res.data.values[i].value
                        }
                    }
                    values.push(count);
                });
                var option = utils.createChartBaseOption('告警级别数据',
'50px',null,null,null,leavels);
                option.xAxis.boundaryGap = true;
                option.tooltip= {
                    show: true,
                    formatter: "{b}:{c}"
                };
                option.series= [{
                    name: '告警数量',
                    type: 'bar',
                    label: {
                        normal: {
                            show: true,
                            position: 'top',
                            formatter: '{c}',
                            textStyle: {
                                color: 'white' //color of value
                            }
                        }
                    },
                    itemStyle: {
                        normal: {
                            show: true,
                            color: function(params) {
                                let num = colorArray.length;
                                return colorArray[params.dataIndex % num]
                            },
                            barBorderRadius: 20,
```

```
                        borderWidth: 0,
                        borderColor: '#333',
                    }
                },
                barWidth: '10%',
                barGap: '0%',
                barCategoryGap: '50%',
                data: values
            }];

            // 使用刚指定的配置项和数据显示图表。
            myChart.setOption(option);
            this.charts['leavel'] = myChart;
        },
        /**
         * 初始化 cpu 图表
         * @param {Object} res
         */
        initTypeChart: function(res) {
            var myChart = this.$echarts.init(document.getElementById('typechart'));

            var colorList = [
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(3, 251, 71, 1)' },
                    { offset: 1, color: 'rgba(19, 218, 140, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(36, 250, 173, 1)' },
                    { offset: 1, color: 'rgba(16, 202, 151, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(33, 245, 219, 1)' },
                    { offset: 1, color: 'rgba(19, 201, 183, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(37, 250, 245, 1)' },
                    { offset: 1, color: 'rgba(11, 190, 204, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(48, 220, 243, 1)' },
                    { offset: 1, color: 'rgba(8, 183, 231, 1)' },
                ]),
                new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                    { offset: 0, color: 'rgba(23, 158, 221, 1)' },
```

```
                        { offset: 1, color: 'rgba(8, 150, 231, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(57, 105, 250, 1)' },
                        { offset: 1, color: 'rgba(41, 85, 237, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(98, 85, 255, 1)' },
                        { offset: 1, color: 'rgba(0, 102, 255, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(150, 71, 254, 1)' },
                        { offset: 1, color: 'rgba(92, 31, 228, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(205, 100, 250, 1)' },
                        { offset: 1, color: 'rgba(149, 70, 254, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(242, 10, 247, 1)' },
                        { offset: 1, color: 'rgba(171, 28, 221, 1)' },
                    ]),
                    new echarts.graphic.LinearGradient(0, 0, 0, 1, [
                        { offset: 0, color: 'rgba(247, 10, 144, 1)' },
                        { offset: 1, color: 'rgba(176, 1, 180, 1)' },
                    ]),
                ];
                var categorys = [];
                var values = [];
                res.data.types.forEach(function(item, index) {
                    categorys.push(item.label);
                    var count = 0;
                    for (var i = 0; i < res.data.values.length; i++) {
                        if (res.data.values[i].type == item.value) {
                            count += res.data.values[i].value
                        }
                    }
                    values.push(count);
                });
                var option = utils.createChartBaseOption('告警类型分布',
'50px',null,null,null,categorys);
                option.xAxis.boundaryGap = true;
                option.series= [
                    {
```

```
                        type: 'bar',
                        data: values,
                        barWidth: '10%',
                        itemStyle: {
                            normal: {
                                color: function (params) {
                                    return colorList[params.dataIndex];
                                },
                                barBorderRadius: [30, 30, 0, 0],
                                shadowBlur: 4,
                            },
                        },
                        label: {
                            normal: {
                                show: true,
                                position: ['-10', '-20'],
                                distance: 1,
                                formatter: ' {a|{c}%}',
                                rich: {
                                    a: {
                                        color: '#fff',
                                        align: 'center'
                                    }
                                }
                            }
                        }
                    }
                ];

                // 使用刚指定的配置项和数据显示图表。
                myChart.setOption(option);
                this.charts['type'] = myChart;
            }
        }
    }
```

九、实例总结

本实例通过 Vue+Echarts 实现了软件系统运行监控界面，可是使用此界面部署到系统中，用于监控在线系统的各项指标运行状况及各资源使用情况，以便判断当前系统运行是否正常。