可视化实战

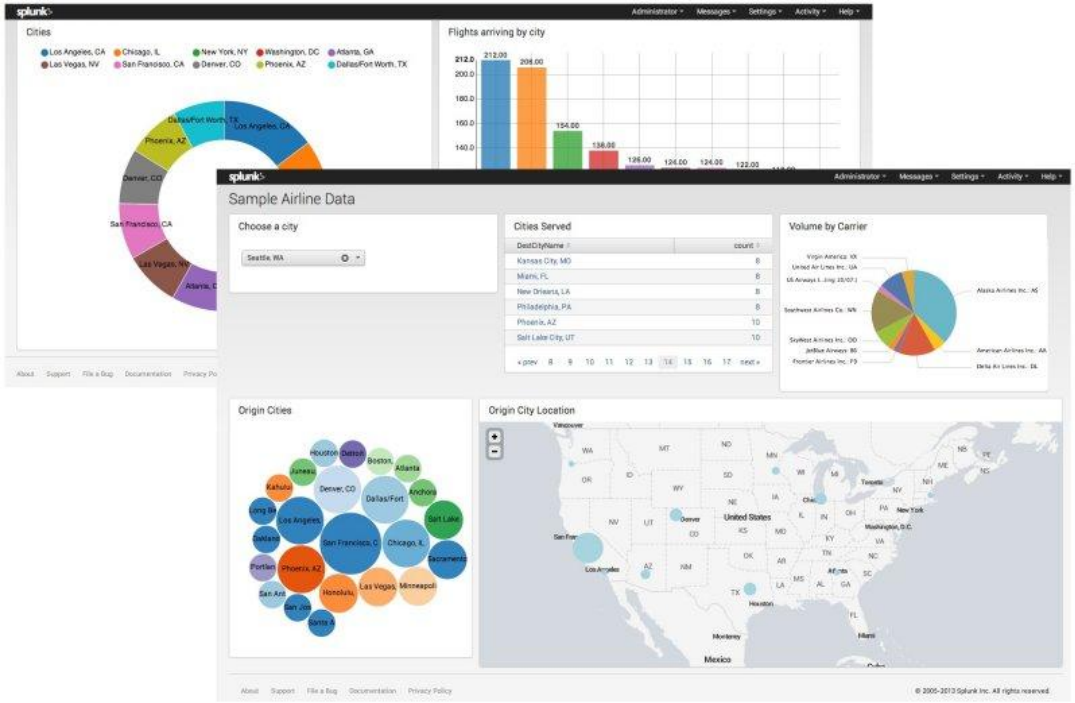
基于MEAN.JS的可视化系统研发

翁荻

浙江大学 计算机辅助设计与图形学国家重点实验室
http://cppmesh.net

可视化与Web技术

无需下载



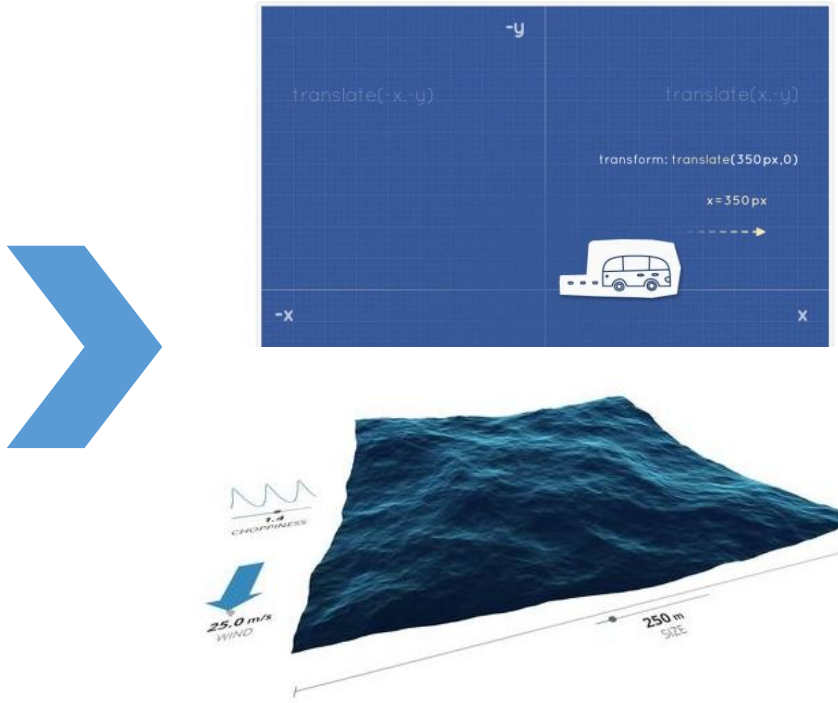
直接运行在浏览器中的可视化系统

多平台支持

从大屏幕到手机：自适应的可视化系统

良好的交互响应性

Empno	Marks			
	Sec A	Sec B	Sec C	Total
1	1	2	3	6
2	4	5	6	15
3	7	8	9	24



借助新兴Web技术实现的交互式可视化

对开发人员要求高

Web开发的框架种类繁多

主流浏览器仅支持JavaScript

如果只用一门语言...

MEAN.JS框架介绍

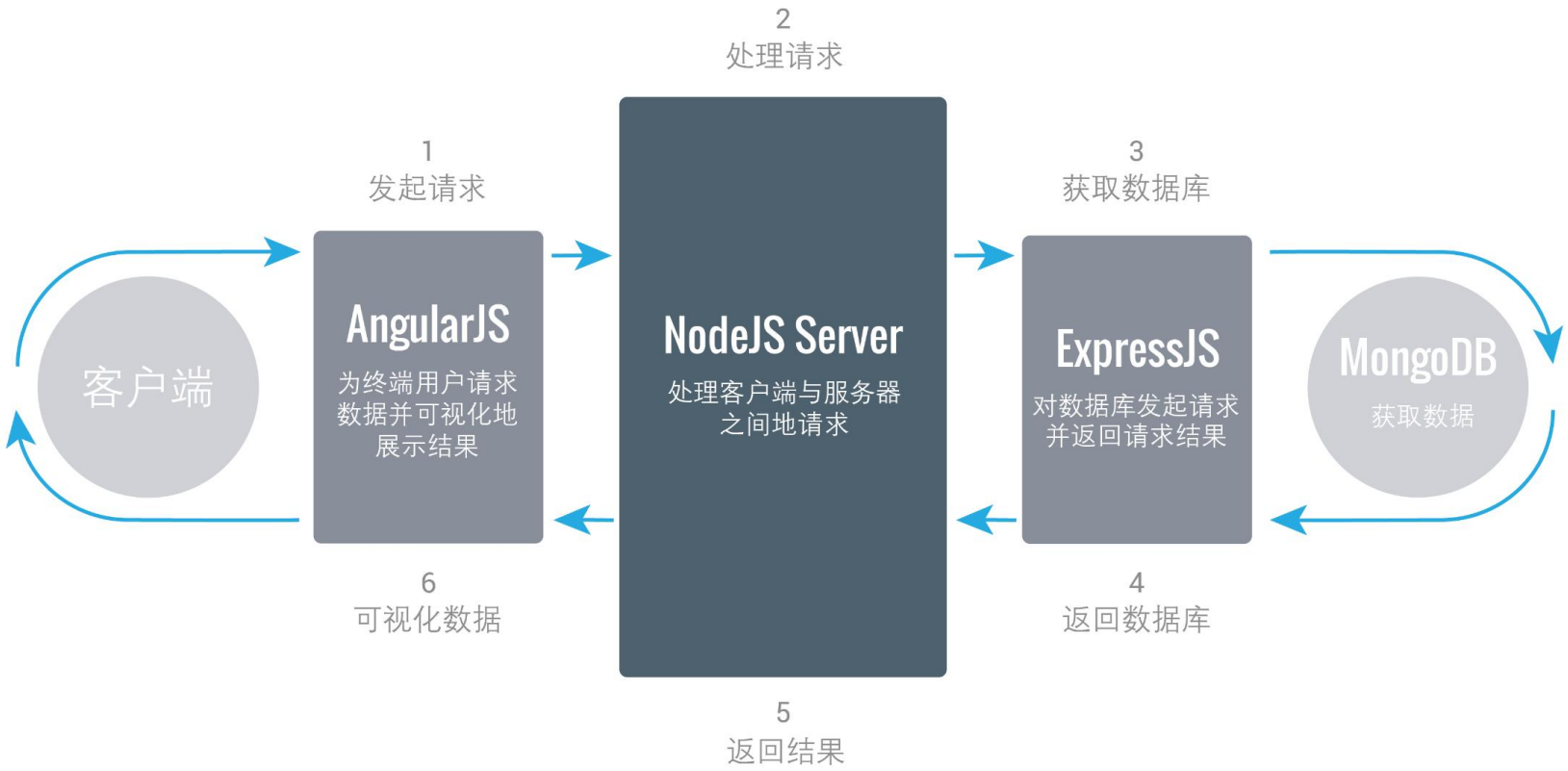
Introduction to MEAN.JS Framework

1

MEAN.JS

MEAN.JS的四大组件

搭建一个可视化系统


2
处理请求
1
发起请求
3
获取数据库
客户端
AngularJS
为终端用户请求数据并可视化地展示结果
NodeJS Server
处理客户端与服务器之间地请求
ExpressJS
对数据库发起请求并返回请求结果
MongoDB
获取数据
6
可视化数据
4
返回数据库
5
返回结果

可视化系统中的数据存储

数据库系统种类如此之多.....

为什么我们选择MongoDB作为可视化系统的数据存储？

MongoDB是NoSQL数据库

数据来源：http://goo.gl/QFQxov

	数据模型	性能	可拓展性	灵活性	复杂性	功能性
NoSQL 数据库	键值存储	高	高	高	低	几乎没有
	基于列的存储	高	高	一般	低	比较小
	面向文档的存储（MongoDB在这里）	高	高	高	低	较低
	图数据库	可变	可变	高	高	基于图论
	关系型数据库	可变	可变	低	一般	关系运算

可视化系统存储的数据结构非常灵活

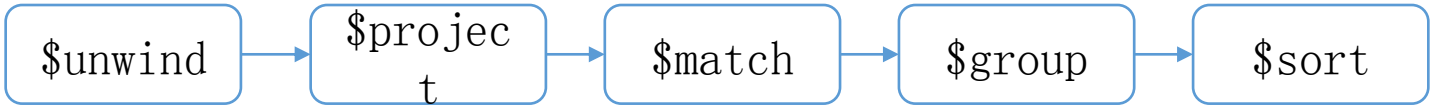
MongoDB的文档结构丰富

MongoDB可以执行复杂的查询

富查询	• 查询Paul拥有的汽车 • 查询所有于1970-1980年间被制造的汽车的车主
地理查询	• 寻找Trafalgar Sq. 5公里内所有的车主
文本查询	• 寻找所有描述中包含真皮座椅的车辆
聚合查询	• 计算Paul拥有车辆的平均价值
Map Reduce	• 拥有车辆的颜色按地理位置是如何分布的（紫色车辆在中国更受欢迎吗？）

```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

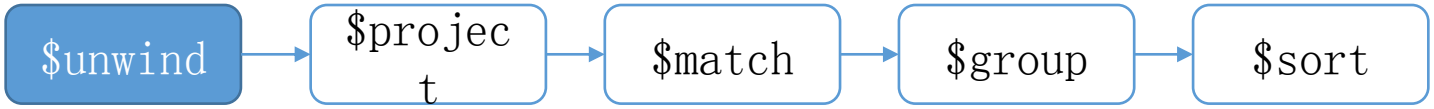
一个例子：MongoDB的聚合查询




```
{
  first_name: ‘Paul’,
  surname: ‘Miller’,
  city: ‘London’,
  location: [45.123,47.232],
  cars: [ … ]
}
```

查询1970年到1980年间，
所有不同品牌的车辆价值总和，并从大到小排序

一个例子：MongoDB的聚合查询

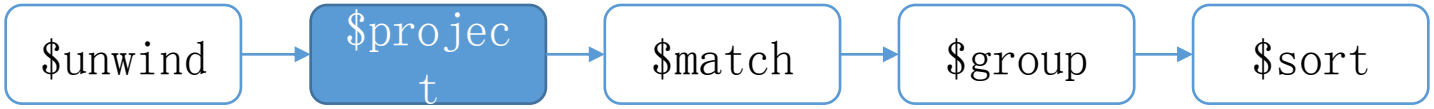



```
{cars: [{ A }, { B }, { C }], … }
{cars: [{ D }, { E }], … }
{cars: [{ F }, { G }], … }
```

```
{cars: { A }, … }
{cars: { B }, … }
{cars: { C }, … }
{cars: { D }, … }
{cars: { E }, … }
{cars: { F }, … }
{cars: { G }, … }
```

用$unwind操作符展开cars数组

一个例子：MongoDB的聚合查询




```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location: [45.123,47.232],
  cars: {
    model: 'Bentley',
    year: 1973,
    value: 100000, … }
}
```

```
{
  model: 'Bentley',
  year: 1973,
  value: 100000
}
```

用$project操作符映射需要操作的域

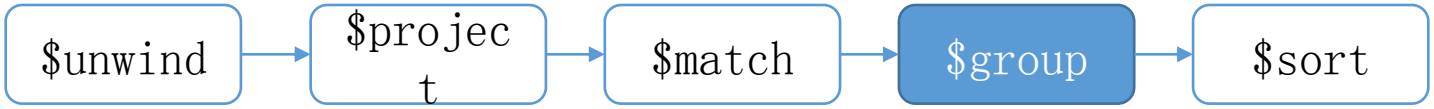
一个例子：MongoDB的聚合查询


```
{ year: 1973, … }
{ year: 1976, … }
{ year: 1966, … }
{ year: 1983, … }
```

```
{ year: 1973, … }
{ year: 1976, … }
```

用$match操作符筛选文档

一个例子：MongoDB的聚合查询



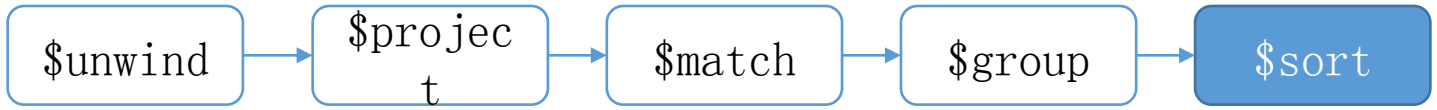

```
{
  model: 'Bentley',
  year: 1973,
  value: 100000 }
```

```
{
  model: 'Bentley',
  year: 1977,
  value: 150000 }
```

```
{
  _id: 'Bentley',
  sum: 250000 }
```

用$group操作符对文档分类并聚合

一个例子：MongoDB的聚合查询




```
{ model: 'Bentley',
  sum: 250000 }
{ model: 'Rolls Royce',
  sum: 400000 }
{ model: 'Porsche',
  sum: 150000 }
```

```
{ model: 'Rolls Royce',
  sum: 400000 }
{ model: 'Bentley',
  sum: 250000 }
{ model: 'Porsche',
  sum: 150000 }
```

用$sort操作符对文档的域降序排序

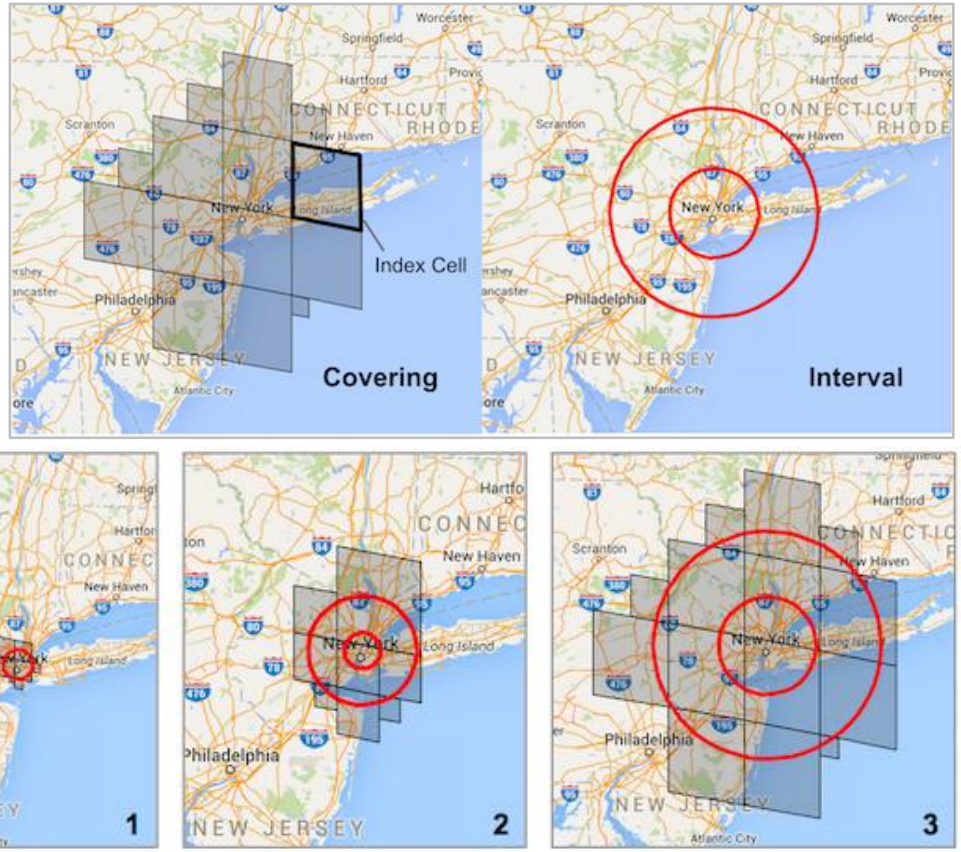
MongoDB与地理信息可视化

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",   经度      纬度
    "coordinates": [-120.0910, 30.3045]
  },
  "properties": {
    "name": "Hack Reactor"
  }
}
```

MongoDB使用GeoJSON来存储地理信息

MongoDB的2dsphere索引

- 递归切割地图
- 利用索引格检索地理位置
- https://goo.gl/JiprVW



MongoDB vs. MySQL

MongoDB	RDBMS
Collection	Table
Document	Row
Index	Index
Embedded Document	Join
Reference	Foreign Key

可视化系统中的服务器搭建

可视化系统服务器的需求

- 方便快捷地处理数据库的数据
  - 与MongoDB对接
- 承受大量的并发连接
  - 大量用户同时使用可视化系统
- 进行复杂的数据计算
  - 在线查询、分析与更新数据
- 简单易用，开发难度低
  - 将工作的重点转移到可视分析上

可视化系统服务器的需求

- 方便快捷地处理数据库的数据
  - 与MongoDB对接
- 承受大量的并发连接
  - 大量用户同时使用可视化系统
- 进行复杂的数据计算
  - 在线查询、分析与更新数据
- 简单易用，开发难度低
  - 将工作的重点转移到可视分析上

Node.js与MongoDB


```
{ model: 'Rolls Royce',
  sum: 400000 }
```

```
{ model: 'Rolls Royce',
  sum: 400000 }
```

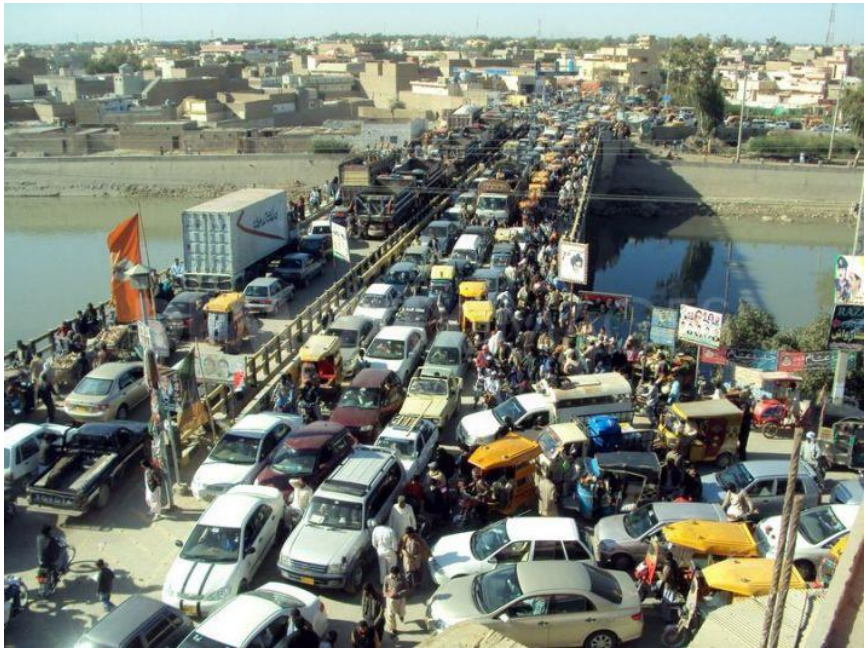
```
{ model: 'Rolls Royce',
  sum: 400000 }
```

保持一致的数据结构

可视化系统服务器的需求

- 方便快捷地处理数据库的数据
  - 与MongoDB对接
- 承受大量的并发连接
  - 大量用户同时使用可视化系统
- 进行复杂的数据计算
  - 在线查询、分析与更新数据
- 简单易用，开发难度低
  - 将工作的重点转移到可视分析上

如何服务并发的请求



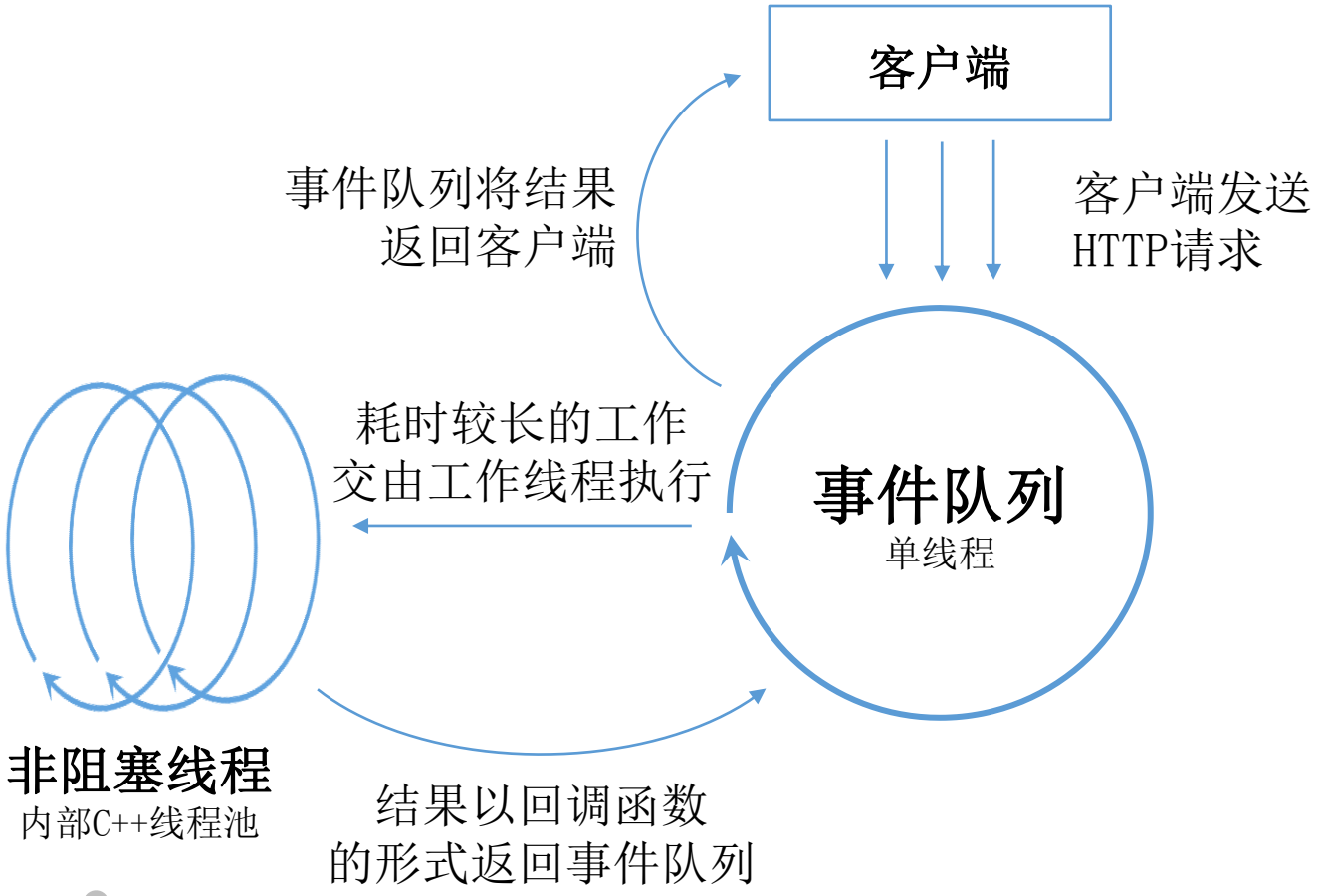
多线程？

复杂的线程同步机制

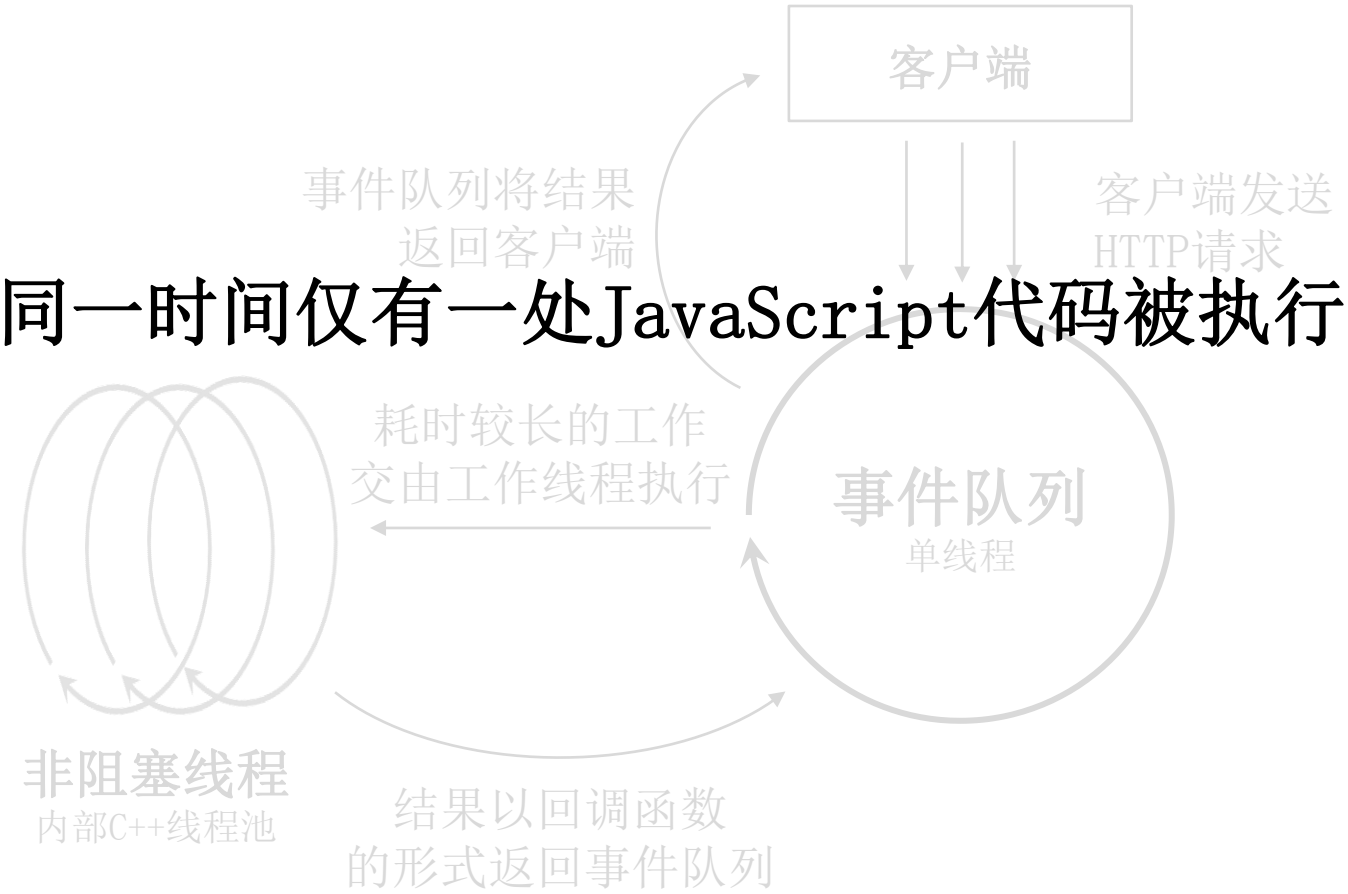
- Spinlock
- Mutex
- Semaphore
- Condition
- …

有没有更简单的解决方案？

Node. js单线程异步模型



Node. js单线程异步模型



Node. js单线程异步模型

同一时间仅有一处JavaScript代码被执行

~~线程死锁、饥饿~~

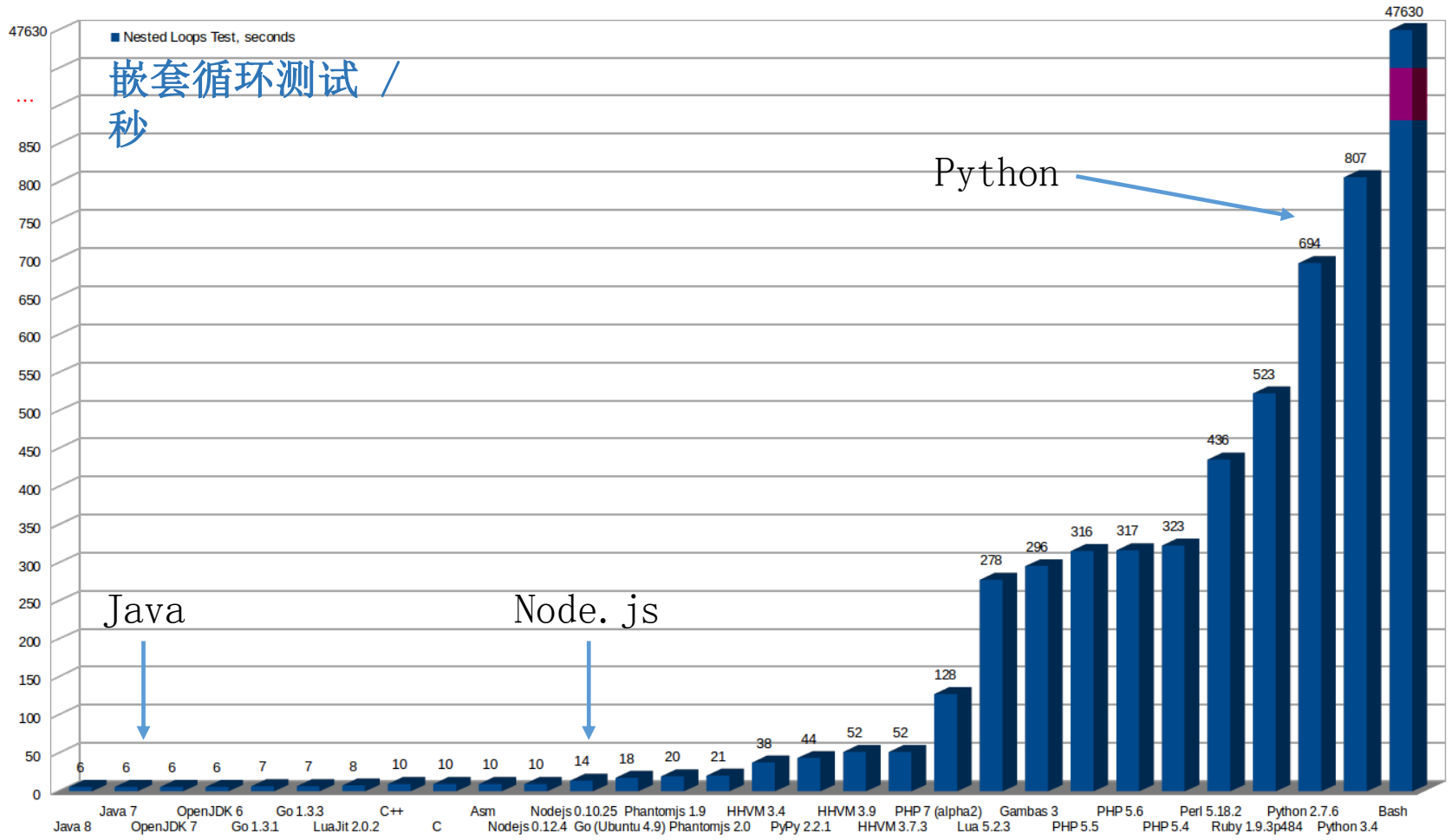
~~线程之间资源共享~~

~~多线程编程难度高~~

可视化系统服务器的需求

- 方便快捷地处理数据库的数据
  - 与MongoDB对接
- 承受大量的并发连接
  - 大量用户同时使用可视化系统
- 进行复杂的数据计算
  - 在线查询、分析与更新数据
- 简单易用，开发难度低
  - 将工作的重点转移到可视分析上

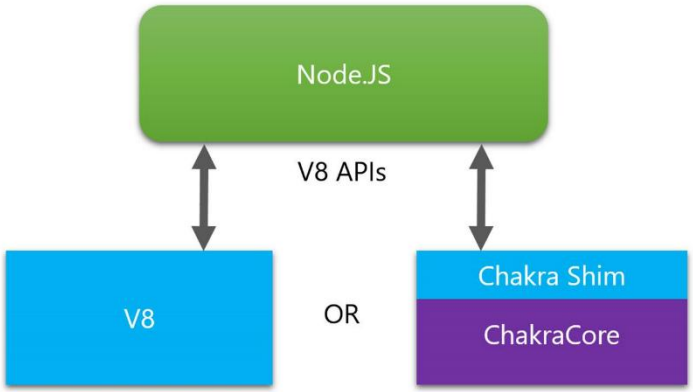
JavaScript性能



还可以更快一点吗？

Node.js的C++ Addon

- 提供可被JavaScript调用的C++代码
- 使用C++直接操作JavaScript对象
- 利用多线程提高运算效率

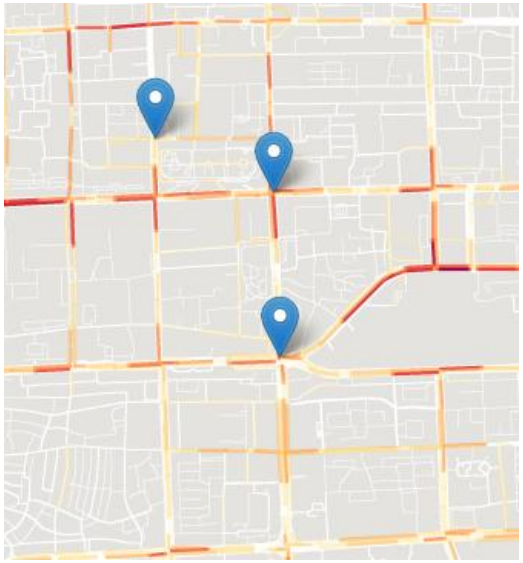


一个例子：数据挖掘模块

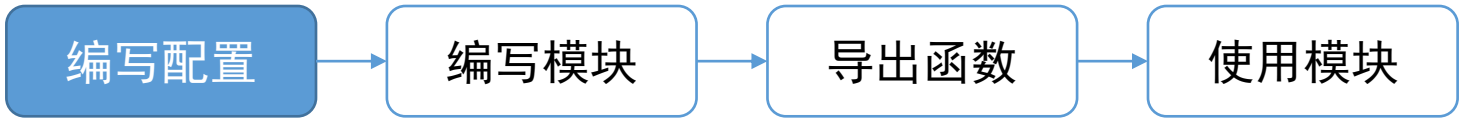
编写配置 → 编写模块 → 导出函数 → 使用模块

问题概括：选出K个位置，使得覆盖的车辆轨迹权值和最大

- 每条车辆轨迹都有权值w_i
- 每个位置都有花费c_j



一个例子：数据挖掘模块



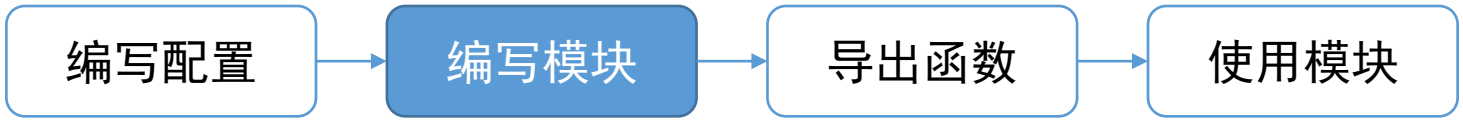

```
{
  "targets": [
    {
        "target_name": "calculator",
        "sources": [ "solver.cpp", "calculator.cpp"]
    }]
}
```

模块名字

模块包含的C/C++源文件

保存为binding.gyp

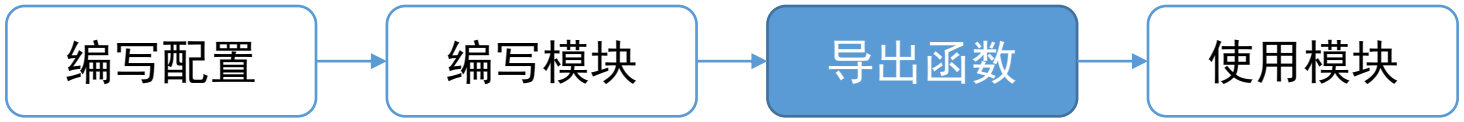
一个例子：数据挖掘模块



利用V8 API编写C/C++源码

参考文档：https://nodejs.org/api/addons.html

一个例子：数据挖掘模块



使用node-gyp编译原生模块

初始化

```
$ node-gyp configure
```

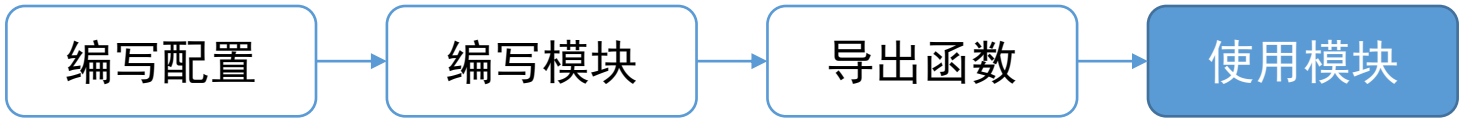
编译

```
$ node-gyp build
```

编写JavaScript接口
calculator.js

```
exports = module.exports =
  require('./build/Release/calculator');
```

一个例子：数据挖掘模块



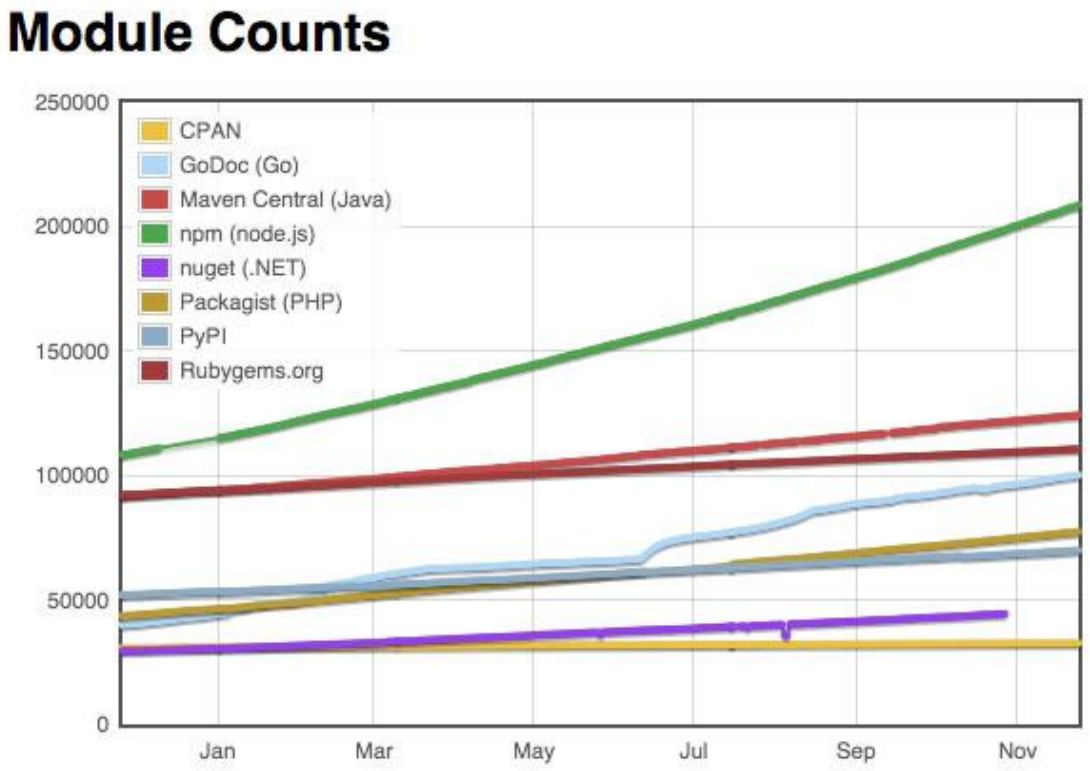

```
var calculator = require('./calculator');
console.log(calculator.getAnswer(Data));
```

在Node. js中使用模块

可视化系统服务器的需求

- 方便快捷地处理数据库的数据
  - 与MongoDB对接
- 承受大量的并发连接
  - 大量用户同时使用可视化系统
- 进行复杂的数据计算
  - 在线查询、分析与更新数据
- 简单易用，开发难度低
  - 将工作的重点转移到可视分析上

庞大的开发社区



提供超过20万个功能丰富的Node.js模块

基于HTML模板的渲染

浏览器访问的URL

被渲染的模板

```
app.get('/cars',
  function (req, res) {
    res.render('cars_view', {
      cars: mycars
    });
  });
```

指定模板中使用的JS变量

route.js

从另外一个模板继承

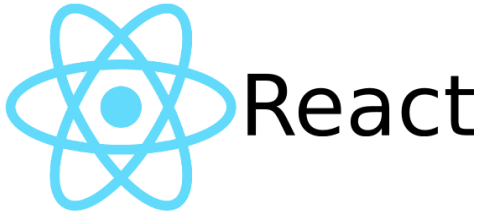
```
{% extends 'layout.html' %}

{% block content %}
  <ul>
    {% for car in cars %}
      <li>car.name</li>
    {% endfor %}
  </ul>
{% endblock %}
```

遍历给定的数据

cars_view.html

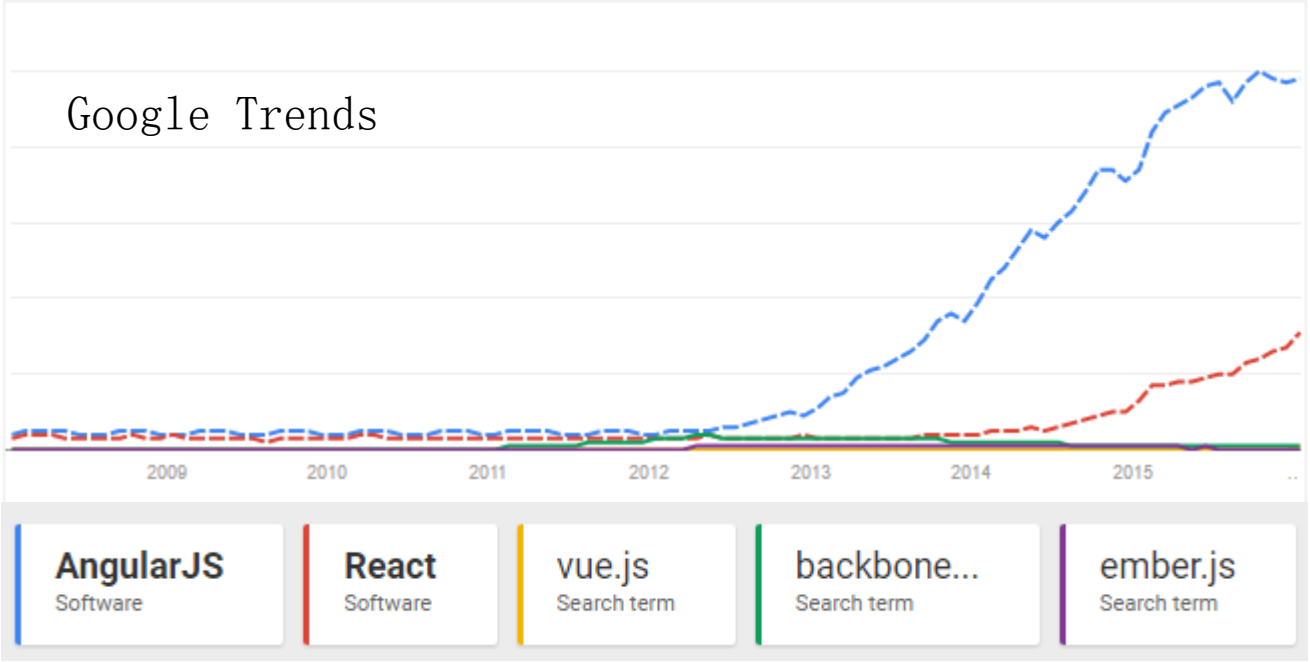
可视化系统中的前端设计

前端框架



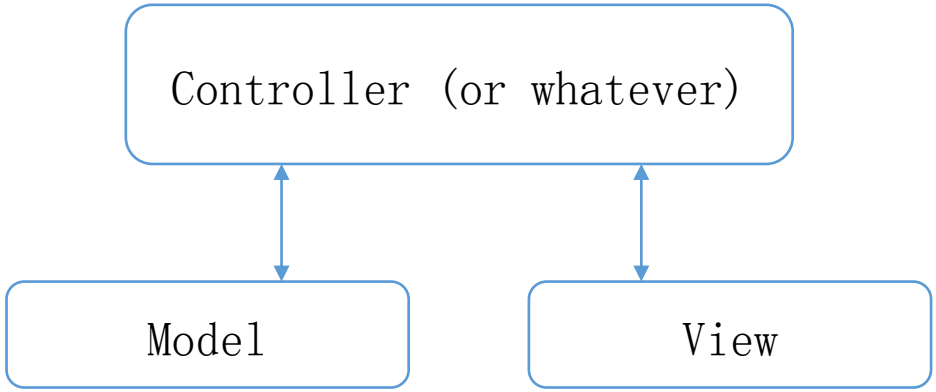
帮助开发人员快速搭建Web应用

为什么选择AngularJS



因为它很流行

Angular是...

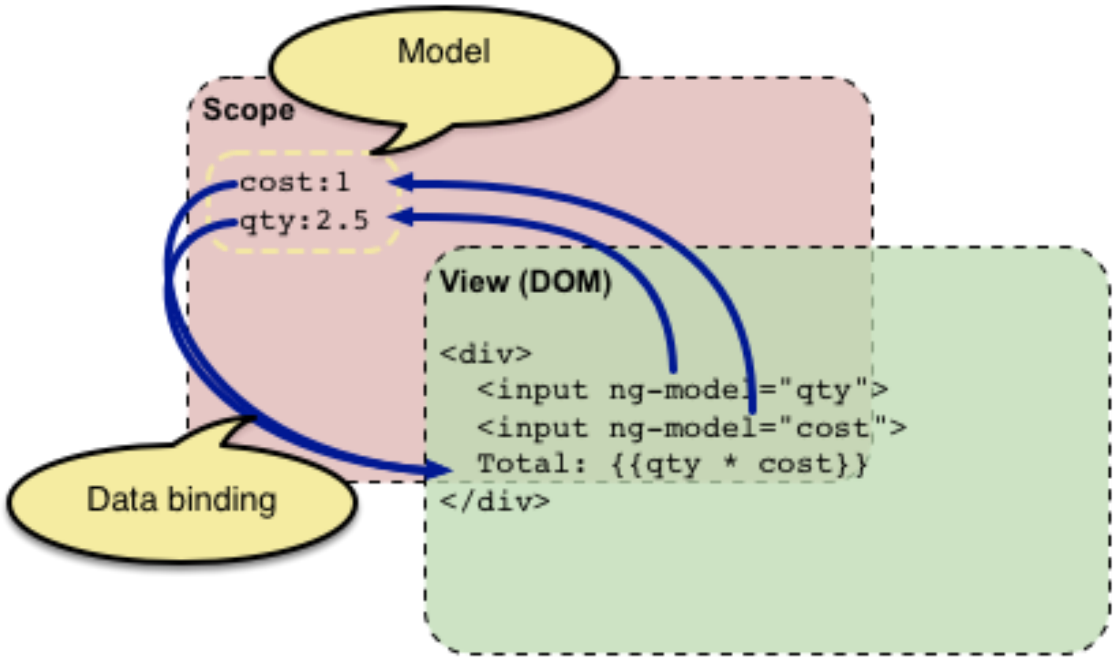


Google开发的一款MVW框架

Angular中的View

```
<!doctype html>
<title>Angular Test</title>
<body ng-app>   为子元素初始化AngularJS
  <div ng-controller="MyController"> 定义DOM元素的controller
    <ul>
      <li ng-repeat="car in myCars"> 遍历model中的数组
        {{car.model}}: {{car.value}}
      </li>   用类似JS的表达式双向绑定模型中的变量
    </ul>
  </div>
</body>
```

双向数据绑定



AngularJS通过digest循环实现双向数据绑定

Angular中的Controller与Model

定义controller所属的模块

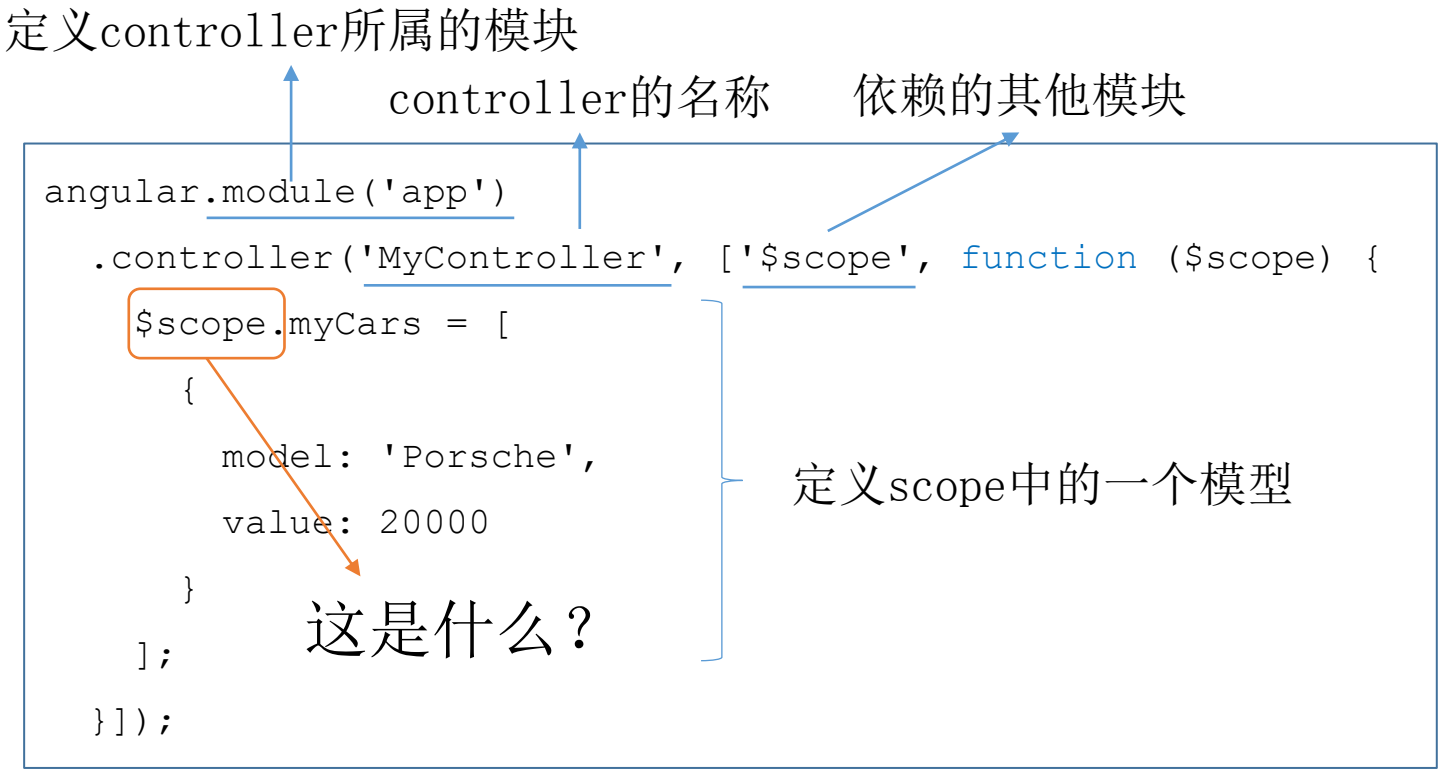
controller的名称

依赖的其他模块

```
angular.module('app')
  .controller('MyController', ['$scope', function ($scope) {
    $scope.myCars = [
      {
        model: 'Porsche',
        value: 20000
      }
    ];
  }]);
```

定义scope中的一个模型

Angular中的Controller与Model



作用域（Scope）


<!DOCTYPE html>
<html ng-app class="ng-scope">
$scope
<head>…</head>
<body>

继承scope成员

Hello World!

$scope
name='World'

$scope
names=[...]

<!-- ngRepeat: name in names -->
<li ng-repeat="name in names" class="ng-scope ng-binding">
Igor

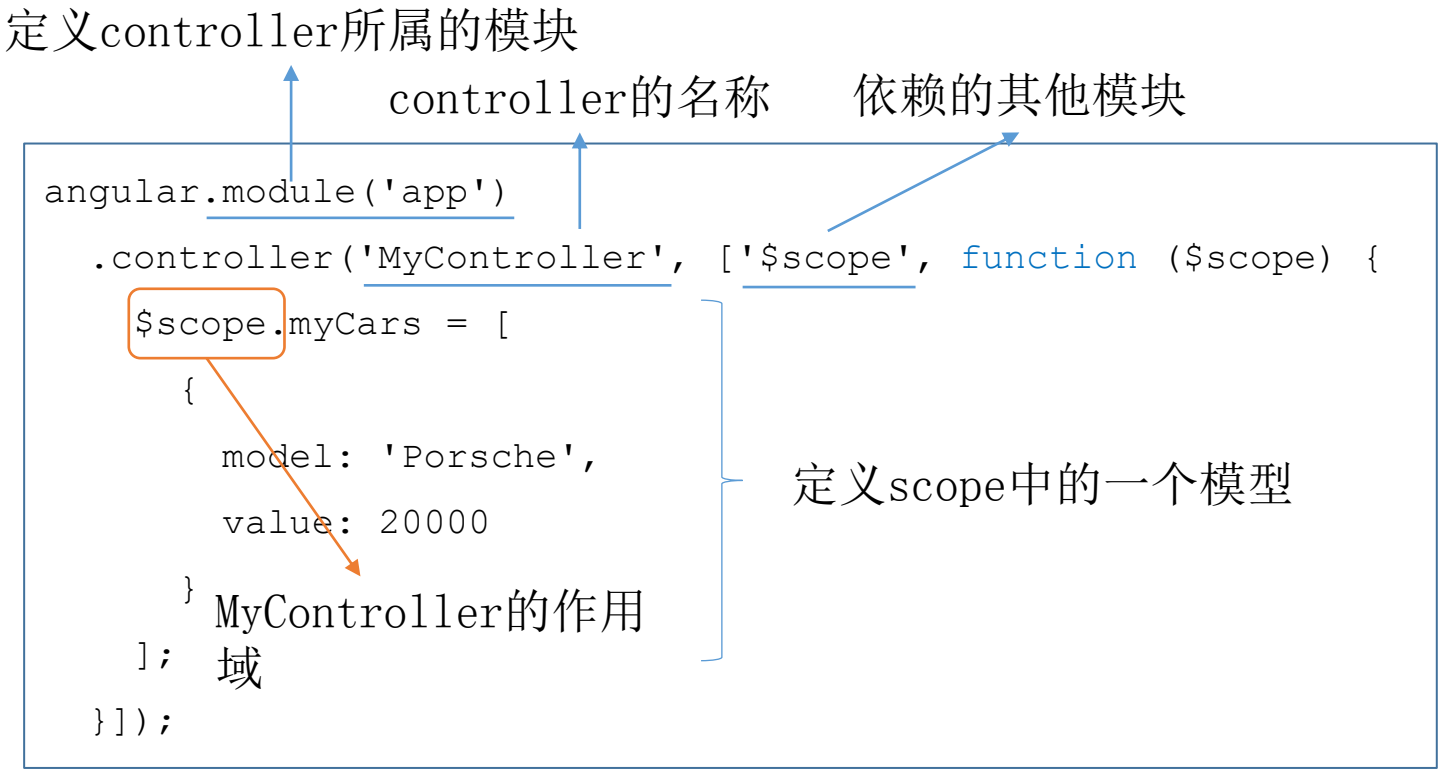
$scope
name='Igor'
<li ng-repeat="name in names" class="ng-scope ng-binding">
Misko

$scope
name='Misko'
<li ng-repeat="name in names" class="ng-scope ng-binding">
Vojta

$scope
name='Vojta'

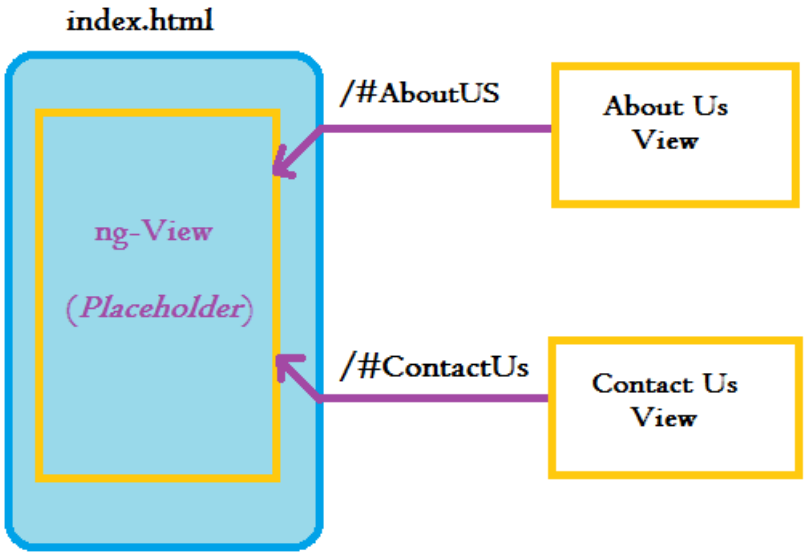
</body>
</html>
scope之间
变量不可见

Angular中的Controller与Model



AngularJS还可以给我们带来...

- 动态模板加载
- 客户端路由
- http://goo.gl/ObCOBA

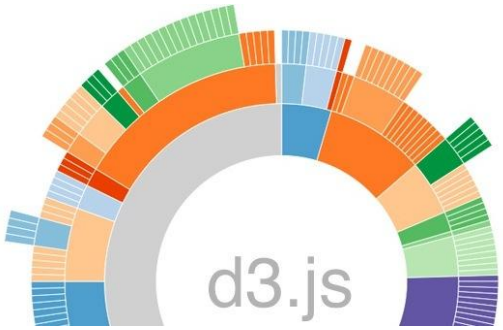


AngularJS如何与可视化相结合呢？

D3.js

- 一套可视化的模块集合
- 简单易用，类似jQuery的语法
- **从JavaScript直接操作DOM元素**

```
d3.select('g')
  .append('circle')
    .attr('cx', 10)
    .attr('cy', 10)
    .attr('r', 5)
    .attr('fill', 'blue');
```



辛辛苦苦分离了几十年的DOM和代码，

一夜就回到解放前了！

用D3.js的方式解决问题

```
angular.module('app')
  .directive('vis', function () {
    return {
      restrict: 'E',
      template: '<svg></svg>',
      link: function (scope, elem) {
        d3.select(elem)
          .append('circle') ...
      }
    };
  });
```

directive的link函数为D3.js提供DOM元素

vis.js (partial)

```
<vis></vis>
```

index.html (partial)

用angular的directive封装D3.js的操作

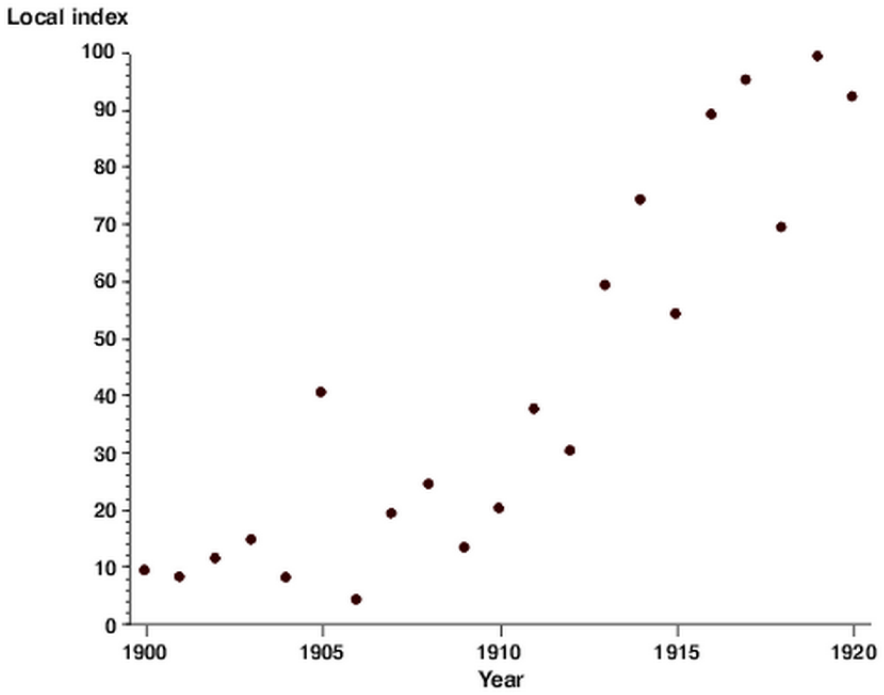
https://goo.gl/wqj7dF

用AngularJS的方式解决问题

- 利用现有的AngularJS可视化组件
  - Angular-nvd3：http://goo.gl/4YXCuy
- 直接在HTML中利用AngularJS的双向数据绑定编写可视化

一个例子：散点图可视化

建立Model → 编写View → 添加交互



一个例子：散点图可视化


```
$scope.model = [
  [1900, 10], [1901, 9], ...
];

$scope.scaleX = x => (x - 1900) / 20 * 640;
$scope.scaleY = y => (y - 0) / 100 * 480
```

在JavaScript代码中定义模型

一个例子：散点图可视化


```
<svg width="640" height="480">
  <circle ng-repeat="point in model"
    cx="{{scaleX(point[0])}}"
    cy="{{scaleY(point[1])}}"
    fill="black">

  ... code to draw axis ...
</svg>
```

利用数据绑定导入需要可视化的数据

一个例子：散点图可视化

- ng-click：绑定点击事件
- ng-mouseenter，ng-mouseleave：绑定鼠标进入/离开事件
- ng-mousemove：绑定鼠标移动事件
- .....

Angular提供丰富的事件监听绑定

更多学习资料

- MongoDB：https://docs.mongodb.com
- Express：http://expressjs.com/en/starter/hello-world.html
- AngularJS：https://docs.angularjs.org/tutorial
- Node.js：https://nodejs.org/dist/latest-v6.x/docs/api/

开发案例：SmartAdP 2

SmartAdP Visual Analytics System

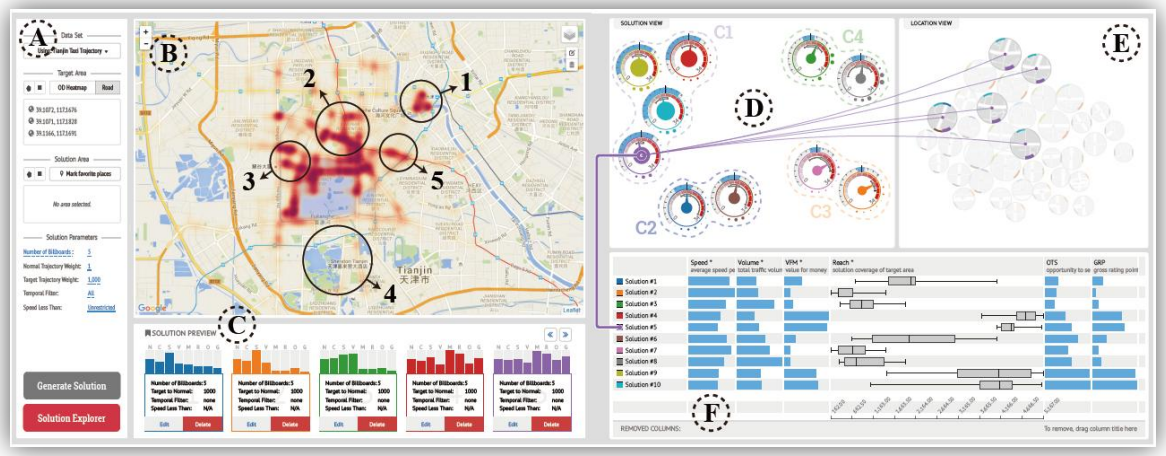
SmartAdP

- 基于大规模出租车轨迹数据
- 用于放置户外大型广告牌的可视分析系统

浙江大學 ZHEJIANG UNIVERSITY

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Microsoft Research 微软亚洲研究院



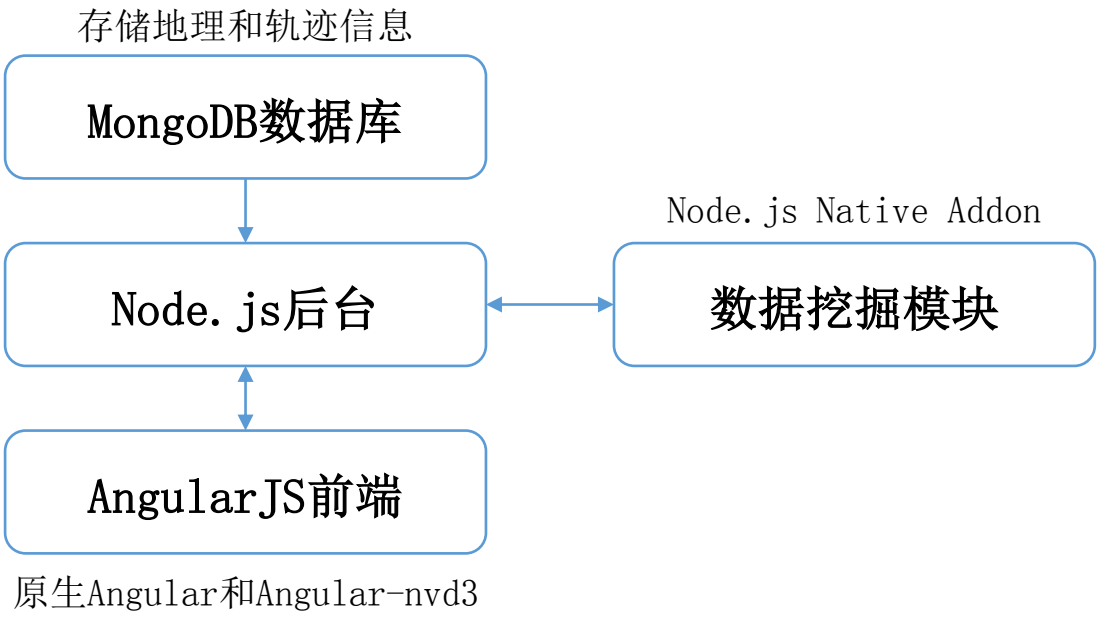
发表文章 SmartAdP: Visual Analytics of Large-scale Taxi Trajectories for Selecting Billboard Locations

Accepted By IEEE VAST 2016 (also appear in TVCG)

广告牌的位置

- 地理位置对于户外广告牌的放置是至关重要的

解决方案



一套结合先进数据挖掘方法和交互式可视化技术的**可视分析系统**

数据规模

- 天津市出租车轨迹数据集
  - 路网数据：133726条道路，99007个顶点
  - GPS轨迹数据： 400多万条轨迹
  - POI数据：154633个地理坐标
- 数据总量：47.3GB
- 分布在640074个文件中

后端数据处理与索引建立

以大规模出租车轨迹数据为例

数据处理：第一次尝试

- 将数据导入MongoDB并做索引
- Python + pymongo
  - 简单易用
  - 需要超过24小时的数据处理时间
  - 字符串处理性能低下

数据处理：第二次尝试

- C++、libbson和MongoDB的C++原生驱动
  - 数个小时的处理时间
  - 连接到MongoDB的网络操作产生阻塞
  - 只有1/12个CPU核心在忙碌

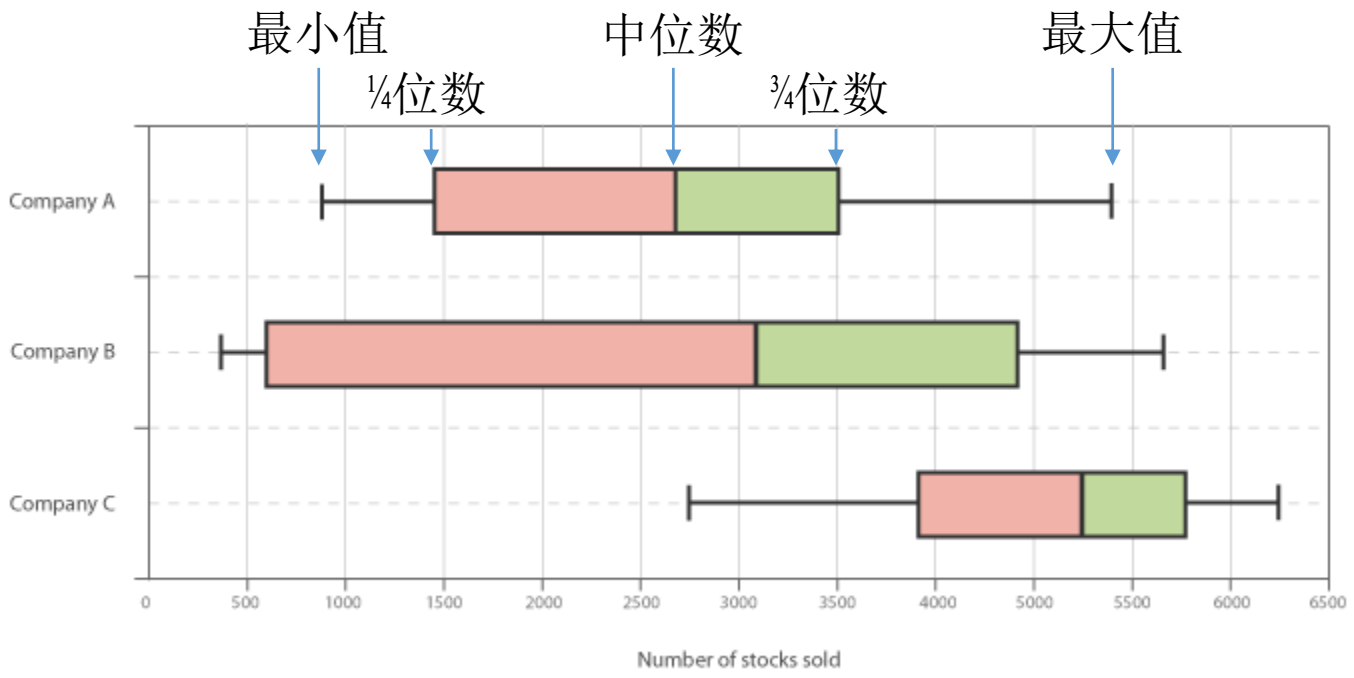
数据处理：最终版

- 独立的MongoDB提交线程、简单线程池和双缓冲机制
  - 扫描并创建文件处理队列
  - 2个可以容纳3000条数据的缓冲区
  - 10个数据处理线程
  - 细粒度的线程锁降低阻塞
  - ~6 min
  - Hash索引服务器

前端可视化的实现

以Boxplot为例

什么是Boxplot



用于呈现数值分布情况的一种可视化

AngularJS的实现

```
<g class="boxplot-values" ng-repeat="row in boxplot.each track by $index"
    style="transform: translate(0, {{row.data.index * rowHeight}}px)">
  <path class="boxplot-line"
    ng-attr-d="M0,{{(0.5 * rowHeight)}}L1,{{(0.5 * rowHeight)}}"
    style="transform: translate({{(row.minimum * boxplot.column.displaySize)}}px, 0
  <path class="boxplot-line"
    ng-attr-d="M0,8l0,{{(rowHeight - 16)}}"
    style="transform: translate({{(row.minimum * boxplot.column.displaySize)}}px, 0
  <path class="boxplot-line"
    ng-attr-d="M0,8l0,{{(rowHeight - 16)}}"
    style="transform: translate({{(row.maximum * boxplot.column.displaySize)}}px, 0
  <rect class="boxplot-half-rect" x="0" y="4" width="1"
```

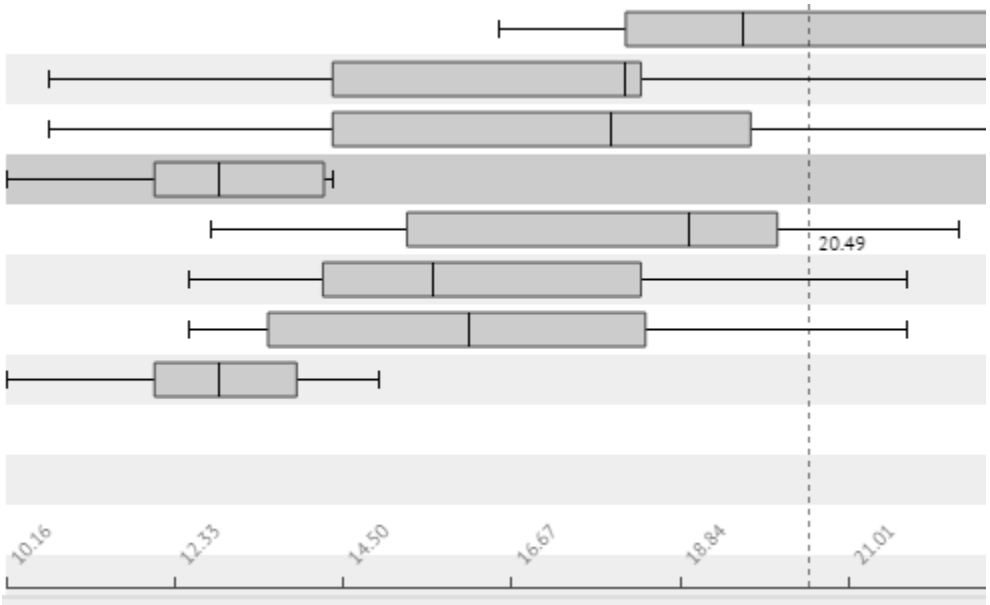
双向绑定transform属性和SVG元素属性

动画效果的实现

```
.lineup-value-bar {
    opacity: 1;
    pointer-events: none;
    transition: transform 0.3s, opacity 0.2s;
    fill: #5FB2E8;
}
```

借助CSS3的transition属性实现动画效果

效果图



视频演示

- 展示SmartAdP可视分析系统的交互功能（1’03″）

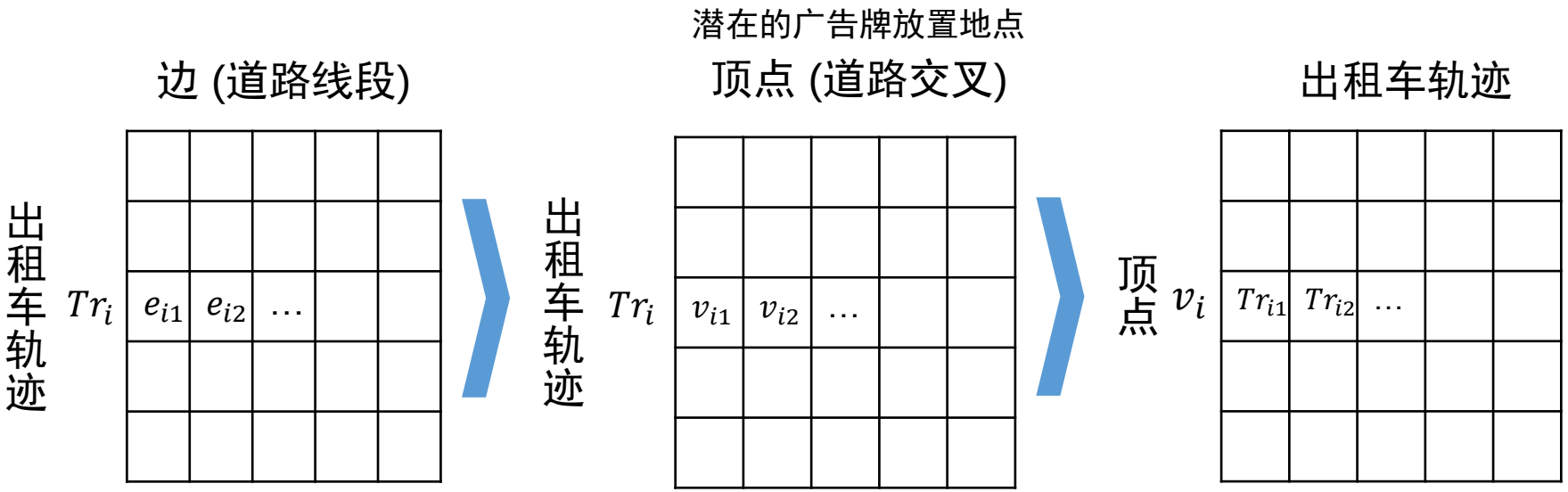
谢谢

E-mail: mystery.wd@gmail.com

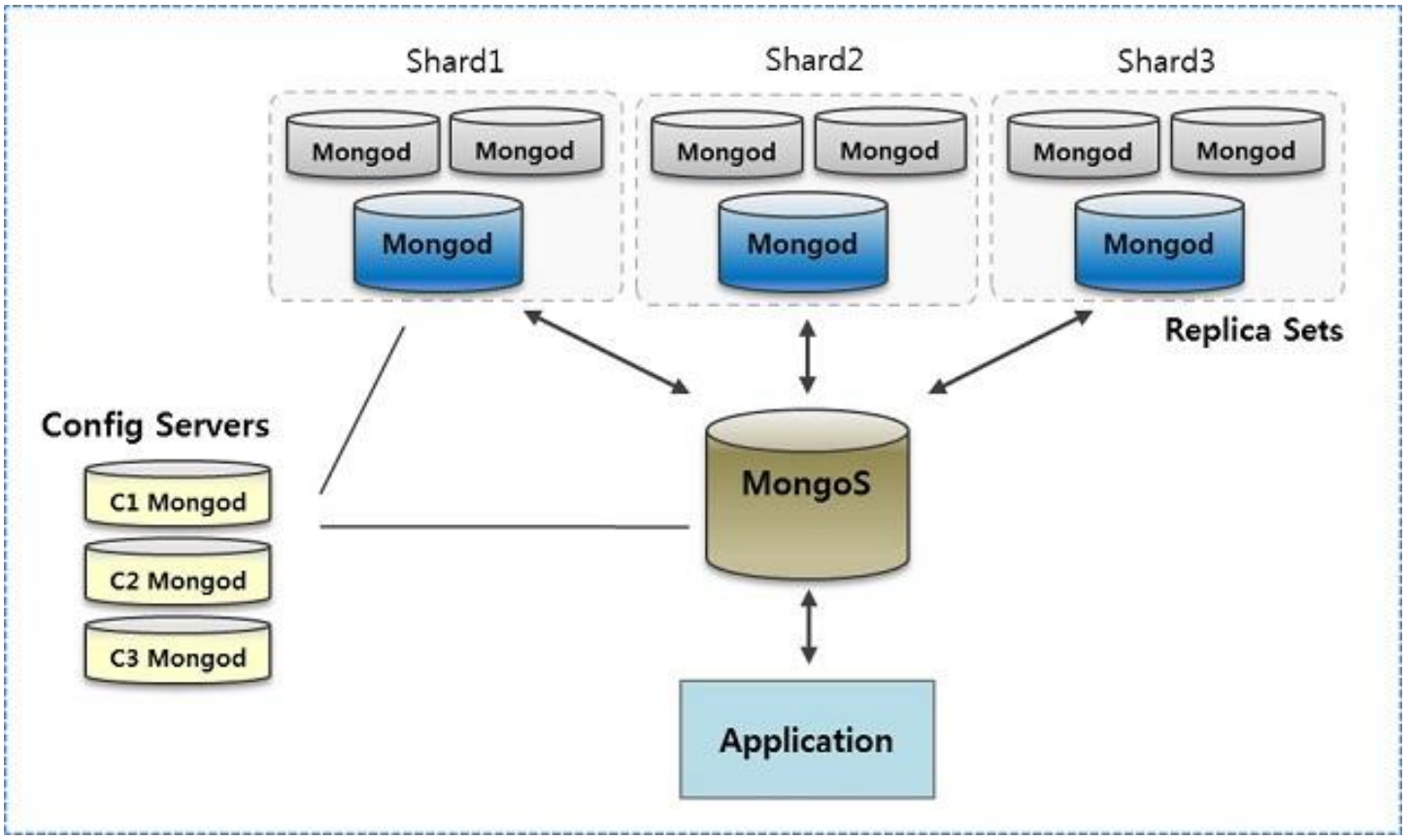
如有问题，欢迎扫描二维码添加加我的微信

数据结构的构建

- 为了从大规模出租车轨迹数据中提取信息并且加速数据挖掘算法，我们建立了三套索引：



通过Shards提高性能



数据挖掘算法

- 问题概括：选出K个位置，使得覆盖的轨迹权值和最大
  - 每条轨迹都有权值w_i
  - 每个位置都有花费c_j
- K-Location Query
  - 选取 & 更新
- T -Budget Constraint Query
  - 利用率

$$U(v)=\frac{\text{current coverage value}}{\text{cost}}$$

快速近似算法性能的下界：

$$1-\frac{1}{e}$$