Scenario 1: Logging

线上运行的系统每天会产生大量的日志，日志里会包含一些错误、警告、及用户行为等各种信息。通常系统会以文本的形式记录日志信息，这样可读性强，从而方便于日常定位问题。但产生大量的日志之后，要想从大量日志里挖掘出有价值的内容，则需要对数据进行进一步的存储和分析。这时我会选择用MangoDB来储存日志条目。因为它有如下几个优点：JSON风格文件的形式，面向文档存储；对任何属性可以索引；丰富的查询；属性易于拓展等。一般的日志包含时间、访问来源、用户、访问目的地址、访问结果、用户使用的系统、浏览器等。我们可以根据需求设置这些字段。每个字段将作为一个属性储存在同一对象里。例如：

{\_id: ObjectId(‘xxxxx123456789’), host: “127.0.0.1”, time: ISODate("2000-10-10T20:55:36Z"), user: “example”, path: “/public/demo.js”, user\_agent: “xxxxxxxxx”, other: {xx: xxx}}

对于允许用户提交日志条目，可以使用简单的静态HTML页面和AJAX。在HTML里，将会提供一个登录界面，只有登陆后用户才能使用系统，这是最基础的。登陆后会有一些子页面分别对应着不同的功能，例如提交日志、查询日志等。在提交日志的页面中，会有一个表单。这个表单包含了不仅包含了日志的基本字段信息，还提供了可自定义字段。比如设置一个添加按钮，点击后AJAX会生成一个新的标签用来记录字段的属性和值。提交后这些自定义的字段会储存在日志对象里的other对象属性里。

查询时，用户们将可以针对每个属性进行查询，并且最多支持两个字段同时查询。例如可以查看用户A的所有日志，也可以查询用户A在某一天的所有日志。同时对于普通用户，只允许其访问基本的日志，只有管理员可以看到所有等级日志信息（例如一些警报等）。

网络服务器可以使用大公司的云服务器，例如亚马逊的。因为他们有好的安全管理能力，通过购买完整的安全服务，我们将不会担心数据丢失或者被大量入侵等问题。

The system running online will generate a large number of logs every day, and the logs will contain various information such as errors, warnings, and user behaviors. Usually, the system records log information in the form of text, which is highly readable and facilitates daily locating problems. However, after a large number of logs are generated, in order to mine valuable content from a large number of logs, further data storage and analysis are required. At this point I would choose to use MongoDB to store log entries. Because it has the following advantages: the form of JSON-style files, oriented to document storage; indexable to any key; rich query; attributes are easy to expand, etc. General logs include time, access source, user, access destination address, access result, system and browser used by the user, etc. We can set these fields as required. Each field will be stored in the same object as a key. E.g:

{\_id: ObjectId('xxxxx123456789'), host: “127.0.0.1”, time: ISODate("2000-10-10T20:55:36Z"), user: “example”, path: “/public/demo.js ”, user\_agent: “xxxxxxxxx”, other: {xx: xxx}}

For allowing users to submit log entries, simple static HTML pages and AJAX can be used. In HTML, a login interface will be provided, and users can use the system only after login, which is the most basic. After logging in, there will be some sub-pages corresponding to different functions, such as submission log, query log, etc. In the page where the log is submitted, there will be a form. This form contains not only the basic field information of the log, but also provides customizable fields. For example, if you set an add button, AJAX will generate a new label to record the new key and values ​​of the field after clicking. After submission, these custom fields will be stored in the “other” object as keys and values in the log object.

When querying, users will be able to query for each attribute, and at most two attributes can be queried at the same time. For example, you can view all logs of user A, or you can query all logs of user A on a certain day. At the same time, ordinary users are only allowed to access basic logs, and only administrators can see all levels of log information (such as some alarms, etc.).

The web server can use the cloud server of a large company, such as Amazon's. Because they have good security management capabilities, by purchasing complete security services, we will not worry about data loss or a large number of intrusions.

Scenario 2：Expense Reports

作为一个完整的web应用程序，它首先要有一个登录的页面。当管理员在后台生成了用户帐号后，用户将使用帐号密码登录web应用。

用户登陆后将会有个提交报销的表单，在这用户将提交一些必要的信息，如：是否报销、报销对象、提交时间、付款方式和金额等。这些数据将储存在MangoDB数据库中，集合中将有这些关键字：id、user、isReimbursed、reimbursedBy、submittedOn、paidOn和amount。用户按要求输入后客户端会检查输入的信息是否符合要求，例如日期格式、有没有特殊字符等。符合要求后所有的信息将会插入数据库中储存。

对于生成PDF文件，我会选择JsPDF库，将通过验证并传入数据库的内容在客户端通过JsPDF库按特定的格式生成一个PDF文件，同时在客户端下载保存这个PDF文件。这里我们要用AJAX获取数据并在页面上生成一个完整的表格，这不一定要显示出来。再写一个function设置PDF页面的大小等数值。接下来就可以生成PDF并保存在本地了。

接下来我们使用AJAX将生成的表格以HTML语句的形式发送给服务端。在服务端，我们可以使用ASP.NET来发送邮件。可以直接利用SmtpMail类发送简单的电子邮件。SmtpMail 类是System.Web.Mail命名空间中最基本的类，它是实现发送电子邮件功能的核心类，无论发送的邮件多么复杂，最终都是通过SmtpMail类中的Send方法发送出去。SmtpMail.Send(from,to,subject,message)。括号中的四个参数分别表示发信人的邮件地址、收件人的邮件地址、邮件主体及邮件内容。这里我们可以通过数据库调用用户名所对应的邮箱地址。然后通过网页读取表单中的mailto、mailfrom、mailsubject和mailBody，然后分别指定给mail对象的 to、from、subject和Body属性，最后调用SmtpMail.Send送出邮件。

关于模版，因为将模板与应用程序代码一起存储在文件中很尴尬，并且这样做会使您的代码变得混乱，我们将向您展示如何使用 Node 的文件系统 API 从单独的文件中读取它们。

As a complete web application, it starts with a login page. After the administrator generates a user account in the background, the user will use the account password to log in to the web application.

After the user logs in, there will be a form for submitting reimbursement, where the user will submit some necessary information, such as: whether to reimburse, object of reimbursement, submission time, payment method and amount, etc. The data will be stored in the MongoDB database and the collection will have these keys: id, user, isReimbursed, reimbursedBy, submittedOn, paidOn and amount. After the user enters as required, the client will check whether the entered information meets the requirements, such as date format, whether there are special characters, etc. All information will be inserted into the database after meeting the requirements.

For generating PDF files, I will choose the JsPDF library, which will generate a PDF file in a specific format through the JsPDF library on the client side through validation and pass the content to the database, and download and save the PDF file on the client side. Here we are using AJAX to fetch the data and generate a complete table on the page, which does not have to be displayed. Write another function to set the size and other values ​​of the PDF page. Next, you can generate a PDF and save it locally.

Next, we use AJAX to send the generated table to the server in the form of HTML statements. On the server side, we can use ASP.NET to send emails. Simple emails can be sent directly using the SmtpMail class. The SmtpMail class is the most basic class in the System.Web.Mail namespace. It is the core class that implements the function of sending e-mails. No matter how complex the sent e-mails are, they are ultimately sent out through the Send method in the SmtpMail class. SmtpMail.Send(from, to, subject, message). The four parameters in brackets represent the sender's email address, the recipient's email address, the email body and the email content, respectively. Here we can call the email address corresponding to the username through the database. Then read the mailto, mailfrom, mailsubject and mailBody in the form through the web page, and then assign to, from, subject and Body attributes of the mail object respectively, and finally call SmtpMail.Send to send the mail.

Regarding templates, since it's awkward to store templates in a file with your application code, and doing so clutters up your code, we'll show you how to read them from a separate file using Node's filesystem API .

Scenario 3: A Twitter Streaming Safety Service

这里我将使用Twitter API中的PowerTrack API。这是通过应用PowerTrack过滤语言来根据各种属性（包括用户属性，地理位置，语言等）匹配推文来实现的。只要设置好相关的位置信息等，就可以拓展到当地以外的区域了。PowerTrack API请求正文中，我们可以设置出publisher、数据格式、从日期、迄今为止、标题以及各种规则。并且最多支持1000条PowerTrack规则。

接下来可以设置一个触发器，令其每30分钟或者一小时检索一个遍。当发现有触发相应警报的推文后，将通过ASP.NET中的SmtpMail发送警报邮件给指定邮箱。同时redis将暂时储存这个推文的所有信息并且安装设定好的规则进行安全分析，从而得出其威胁等级。分析后将威胁信息实时显示。

分析完成后这篇推文的所有信息将储存到历史数据库中。这里也可以采用MangoDB作为数据库，它是一个很适合储存JSON格式的数据库。这里用户可以标记这个记录的调查状态，也可以随时检索所有的历史记录。此外数据库中还将有一个document专门存储日志信息。包括了触发记录、处理记录、查询记录等。对于推文中的所有媒体（图片、URL快照等），将采用GridFS储存到数据库中。同时我们也可以将这些非文本信息保存在云服务器中，例如aws的oss，这样可以大大减轻本地服务器的负担。并且也方便后续拓展到其他地区的警局。

流式事件报告可以利用开源的Kafka开发流式数据平台，对流式数据进行持续、实时的处理和转化，并将结果在整个系统内开放。通过常用的流式处理框架，例如Storm、Samza或Spark Streaming可以很容易得跟流式数据平台整合。这些流式数据处理框架提供了丰富的API接口，可以简化数据转化和处理。

当触发警报的推文完成分析并储存到历史数据库后，redis就可以删除本地的缓存了，这样可以避免系统因数据过多而发生不稳定的现象。

Here I will use PowerTrack API from Twitter API. This is achieved by applying a PowerTrack filter language to match tweets based on various attributes including user attributes, geographic location, language, etc. As long as the relevant location information is set, it can be expanded to areas outside the local area. In the PowerTrack API request body, we can set the publisher, data format, date, date, title and various rules. And supports up to 1000 PowerTrack rules.

Next you can set up a trigger to retrieve it every 30 minutes or an hour. When a tweet that triggers the corresponding alert is found, an alert email will be sent to the specified mailbox through SmtpMail in ASP.NET. At the same time, redis will temporarily store all the information of this tweet and install the set rules for security analysis to obtain its threat level. After analysis, the threat information is displayed in real time.

After the analysis is complete, all information about this tweet will be stored in the historical database. MangoDB can also be used as a database here, which is a database that is very suitable for storing JSON format. Here the user can mark the investigation status of this record and can also retrieve all the history records at any time. In addition, there will be a document in the database dedicated to storing log information. Including trigger records, processing records, query records and so on. All media in tweets (images, URL snapshots, etc.) will be stored in the database using GridFS. At the same time, we can also save these non-text information in the cloud server, such as the oss of aws, which can greatly reduce the burden of the local server. It is also convenient for subsequent expansion to police stations in other regions.

Streaming event reporting can use the open source Kafka to develop a streaming data platform, process and transform streaming data continuously and in real time, and open the results to the entire system. It is easy to integrate with streaming data platforms through common streaming frameworks such as Storm, Samza or Spark Streaming. These streaming data processing frameworks provide rich API interfaces that simplify data transformation and processing.

When the tweets that trigger the alert are analyzed and stored in the historical database, redis can delete the local cache, which can avoid the instability of the system due to too much data.

Scenario 4: A Mildly Interesting Mobile Application

用户登陆后可以上传他们拍摄的有趣的照片，这些图片都将储存到数据库中。我们可以选择使用MangoDB来储存这些图片。MongoDB是一个基于分布式文件存储的数据库，用来储存图片信息非常适合。我们可以用GridFS将图片存入数据库中。GridFS使用两个集合（collection）存储文件。一个集合是chunks, 用于存储文件内容的二进制数据；一个集合是files，用于存储文件的元数据。

毫无疑问这些照片信息里都包含了经纬度等地理信息。那么我们就要从上传的照片里提取经纬度信息来获取定位。这里可以使用exif-js API来获取各种照片信息。使用EXIF.getTag(this, 'GPSLongitude')和EXIF.getTag(this, 'GPSLatitude')可以获得照片储存信息，再通过固定的计算公式就得出经纬度了。在上传到同时，这些经纬度信息也将作为属性存储到数据库中，方便以后查找。

另外，我们还可以将拍摄的图片信息暂时保存在redis里进行一些处理。这里可以使用redis的georadius命令。georadius以给定的经纬度为中心，返回键包含的位置元素当中，与中心的距离不超过给定最大距离的所有位置元素。还可以使用geohash来保存地理位置的坐标。这时我们就得到了用户附近的位置信息，再将这些信息作为关键字在数据库中查找有没有相关的图片。短期快速检索可以在redis里完成，应用运行的时候会将部分信息保存在redis里，当redis里没有找到相关的信息后再通过mangDB进行完整的检索。

After logging in, users can upload interesting photos they have taken, which will be stored in the database. We can choose to use MongoDB to store these images. MongoDB is a database based on distributed file storage, which is very suitable for storing image information. We can use GridFS to store images in the database. GridFS uses two collections to store files. A collection is chunks, which are used to store binary data of the file content; a collection is files, which is used to store the metadata of the file.

There is no doubt that these photo information contains geographic information such as latitude and longitude. Then we need to extract the latitude and longitude information from the uploaded photos to obtain the positioning. Here you can use exif-js API to get various photo information. Use EXIF.getTag(this, 'GPSLongitude') and EXIF.getTag(this, 'GPSLatitude') to obtain the photo storage information, and then obtain the latitude and longitude through a fixed calculation formula. At the same time of uploading, these longitude and latitude information will also be stored in the database as attributes, which is convenient for later search.

In addition, we can also temporarily save the captured image information in Redis for some processing. Here you can use the Georadius command of Redis. Georadius takes the given latitude and longitude as the center, and returns all the position elements contained in the key, and the distance from the center does not exceed the given maximum distance. You can also use Geohash to save the coordinates of the geographic location. At this time, we get the location information near the user, and then use this information as a keyword to find out whether there are related pictures in the database. Short-term fast retrieval can be done in Redis. When the application is running, some information will be saved in Redis. When no relevant information is found in Redis, a complete retrieval will be performed through mangoDB.