一个基于 Web 的轻量级大数据处理与可视化工具

李炎[1,2] **马俊明**[1] **安博**[1,2] **曹东刚**[1,2]
（北京大学高可信软件技术教育部重点实验室 北京 100871）[1]
（北京大学信息科学技术学院 北京 100871）[2]

摘 要 科研人员在日常研究中经常使用 Excel，Spss 等工具对数据进行分析加工来获得相关领域知识。然而随着大数据时代的到来，常用的数据处理软件因单机性能的限制已经不能满足科研人员对大数据分析处理的需求。大数据的处理和可视化离不开分布式计算环境。因此，为了完成对大数据的快速处理和可视化，科研人员不光需要购置、维护分布式集群环境，还需要具备在分布式环境下编程的能力和相应的前端数据可视化技术。这对很多非计算机科班的数据分析工作者来讲是非常困难且不必要的。针对上述问题，本文提出了一种基于 Web 的轻量级大数据处理和可视化工具。通过该工具，数据分析工作者只需通过简单的点击和拖动，便可以在浏览器中轻松地打开大型数据文件（GB 级别）、快速对文件进行定位（跳转到文件某一行）、方便地调用分布式计算框架对文件内容进行排序或求极大值、便捷地对数据进行可视化等操作。本文最后进行了相应的实证研究，证明该解决方案是有效的。

关键词 数据分析，分布式系统，并行计算，数据可视化，大数据

中图法分类号 TP399 **文献标识码** A **DOI**

A Web Based Lightweight Tool for Big Data Processing and Visualization

LI Yan[1,2] Ma Jun-ming[1,2] An Bo[1,2] Cao Dong-gang[1,2]
(Key Lab of High Confidence Software Technologies, Peking University, Beijing 100871, China)[1]
(School of Electronic Engineering and Computer Science, Peking University, Beijing 100871, China)[2]

Abstract Researchers in the daily study often use Excel, Spss and other tools to analyze the data to obtain the knowledge of relevant field. However, with the arrival of large data age, commonly used data processing software due to stand-alone performance constraints cannot meet the needs of researchers for large data analysis and processing. Large data processing and visualization is inseparable from the distributed computing environment. Therefore, in order to complete the rapid processing and visualization of large data, researchers not only need to purchase, maintain a distributed cluster environment, but also need to have the ability to program in a distributed environment and the corresponding front-end data visualization technology. This is very difficult and unnecessary for many non-computer science data analysis workers. In view of the above problems, this paper presents a Web-based lightweight large data processing and visualization tool. Using this tool, data analysis workers can easily open a large data file (GB level) in the browser, quickly locate the file, sort the contents of the file and visualize it through a simple click and drag. At the end of this paper, a corresponding empirical study is carried out to prove that this solution is effective.

Keywords Data Analysis, Distributed System, Parallel Computation, Data Visualization, Big Data
到稿日期： 返修日期：
The National Science and Technology Major Project under Grant No.2016YFB1000105 (国家重点研发计划); the National High Technology Research and Development Project under Grant No.2015AA01A202 (国家高技术研究发展计划 863); the National Natural Science Foundation of China under Grant No.61690201 (国家自然科学基金); the National Natural Science Foundation of China under Grant No.61421091 (国家自然科学基金).

Received 2000-00, Accepted 2000-00.

1 引言

随着大数据时代的到来，科学研究中使用到的数据量越来越大，单机环境的数据分析工具已经不足以支持科研人员的需要。例如，当数据文件大小达到 GB 级别时，Excel 已经不能将文件全部打开，也难以对全部数据进行数据分析和可视化。因此，对于数据分析人员来讲，一个运行在云端、向开发人员提供友好界面的简单易用工具是必须的。例如现有的 Google Sheet[1]、Excel Online[2]等应用都具有这样的特点：数据存储和处理都在云端，表格和图标在前端呈现，用户仅仅需要一个浏览器就可以完成对云端文件的处理。

但是上述的应用对大数据还没有很好的支持。例如在 Excel Online 中，用户只被允许打开小于 5M 的表格文件，这对一部分科研工作者来讲显然是不足的。另一方面，考虑到数据安全的问题，很多中小企业都不选择将关键数据存放在公有云上，而是存放在企业内部的专用云中。针对现有工具的不足，本文设计了一种基于 Web 的大数据处理与可视化方案。该方案具有如下特点：1）用户拥有属于自己的专用虚拟集群，系统按用户需求提供分布式处理环境；2）支持云端自动化的大数据并行化处理；3）支持前端大数据交互可视化，用户仅需一个浏览器即可完成对云端大数据的操作。

完成该方案的实现主要面临如下几个关键技术问题：1）一个云端的数据中心操作系统，该系统能够按需提供分布式处理环境；2）一个基于 1）的云端大数据处理系统；3）一个基于 2）的前端可视化系统。

基于上述方案，本文实现了一个基于 Web 的大数据处理与可视化系统 ，系统后端借助数据中心操作系统的虚拟集群进行数据并行化处理，前端使用 JavaScript 插件完成数据交互与可视化，使数据分析工作者可以通过浏览器方便地查看和处理存储在私有云上的数据。

本文第二节介绍了研究背景和相关工作；第三节具体介绍了系统架构与关键技术；第四节通过实例介绍了该处理方案的应用背景；最后一部分总结。

2 背景与相关工作

文章的相关工作主要分为 3 个部分：第一部分是数据中心操作系统 Docklet；第二部分是数据分析工具；第三部分是可视化技术。

2.1 数据中心操作系统 Docklet

Docklet[1-2]是北京大学软件工程研究所的一个开源项目，遵从新 BSD 协议，代码在 Github[3]上维护。Docklet 的目标是实现一种云化的个人工作区解决方案，使企业可以轻松地虚拟化其数据中心，按需[3]为用户创建个人的虚拟集群，进而为用户提供一个在云端的可定制工作区(Workspace)[4]。用户只需要一个浏览器，即可随时随地访问企业内部自己的工作区，在线进行代码编写、调试运行、数据管理、数据分析、结果可视化等工作。用户通过 Docklet 集成的 Jupyter Notebook[5]访问自己的工作区，通过浏览器完成开发运行等工作。

由于 Docklet 的每个用户都拥有专用的虚拟集群，Docklet 在数据并行化处理上拥有天然的优势，用户也不必担心数据的安全性问题。但目前 Docklet 还尚未支持更为友好的数据分析和可视化方案，用户仍需在云端进行代码编写和调试，无法做到像 Excel 那样通过简单的点击和拖动来分析和可视化数据。

2.2 数据分析工具

当前的数据分析工具主要分为两种，一种是单机环境下的数据分析工具，另一种是网络环境下的数据分析工具。

（1）单机环境下的数据分析工具。

在数据量不是很大的情况下，多数用户会选

[1] https://docs.google.com/spreadsheets

[2] https://office.live.com/start/Excel.aspx

[3] https://github.com/unias/docklet

择在本机进行数据的分析。如使用 Excel 与 Spss，就可以满足大多数人的需求。如果需要更加个性化的数据分析操作，可以选择使用 Python。Python 中有很多适合数据分析的第三方包，如 numpy[6]，Pandas[7]等等。这些数据分析工具操作方便，效率较高。但是受制于单机的环境，在面对数据量较大的情形时，它们常常显得无能为力。

（2）网络环境下的数据分析工具

另一种数据分析工具在云端运行。如用户可以在云服务器厂商租用一组服务器，在云端搭建一个数据分析专用的集群，以期得到更好的性能。但是这种情况下不但需要数据分析人员掌握数据分析相关的编程技术，还需要他们熟悉集群的配置和相应环境的搭建，并不是一种友好的方式。而现有的一些云端数据处理服务也没有达到较高的用户交互水平，用户仍需自己编程，然后在命令行中执行程序或以一个作业的形式提交到集群中。

2.3 可视化技术

可视化技术[18]现在已经非常成熟，无论是单机环境下的可视化工具，还是网络环境下基于 JavaScript 的可视化方案，都能够做出生动美观的图表。

（1）单机下以 Excel 为例，用户在打开数据文件之后，可以方便地进行绘图，求值等操作。但是当文件大小达到一定界限之后，Excel 将不会将整个文件加载，而是选择性地加载前面的一部分。这样的处理策略有两种弊端，一是用户无法对文件被加载之外的数据进行查看或者操作；二是很大的数据量直接被加载到内存中，使计算机负载加重。

（2）网络环境下以 Infogram[8]为例，用户可以直接在浏览器中操作，创建自己的表格，并对表格进行可视化操作，仅需几个按钮即可制作成精美的图表。同时用户可以上传自己的数据文件，在云端对其进行可视化操作。但是当上传文件比较大时，如文件大小达到 500MB，云端的文件打开将会变得非常缓慢，后续的可视化操作更是无法进行。

综上所述，目前的相关工作只关注数据中心操作系统、数据分析和数据可视化的一个或两个方面，用户友好型的数据分析技术和可视化技术尚未与数据中心操作系统相结合。通过充分的调研，在近年的学术会议、期刊以及现有产品中，本工作首先将三者相结合。

3 系统架构与关键技术

本文设计的大数据处理方案基于 Jupyter Notebook，通过一个 Jupyter Notebook 的插件来实现。服务器端使用 Python 的 Web 开发框架 Tornado[9]，数据的并行计算部分使用 Spark[10]与 MPI[11]来实现；浏览器端的交互界面遵循 Jupyter Notebook 使用的前端框架——RequireJS[17]，前端的表格显示使用 JavaScript 插件 Handsontable[12]来实现，图表（柱状图，折线图等）显示通过 Python 第三方包 matplotlib 以及 JavaScript 插件 heatmap.js 来实现。系统的架构图如图 1 所示。

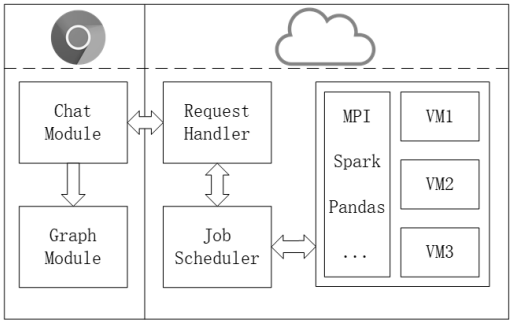

图 1 系统整体架构

系统的主体主要包含如下四个模块：前端表格显示模块、前端图表显示模块、后端请求处理模块、后端并行计算模块。本系统前端采用了 JS 插件 Handsontable。系统的前端通过 JS 对 Jupyter Notebook 的文件列表界面进行修改，将表格文件的打开链接绑定到本插件设计的 URL 处，通过 POST 请求发送到 server 端。后端处理完毕之后将表格数据传回前端，由 Handsontable 负责渲染显示。后端遵循 Tornado 的 Web 开发模式，请求处理由一个

RequestHandler 来处理。前端的通过 AJAX 向后端发送相应请求，并异步等待后端的返回值。

在具体实现中，主要解决了如下三个技术问题：如何快速定位到文件的某一行；如何利用 Docklet 集群的优势，在后端对数据进行并行化处理；如何进行有效的可视化。

3.1 如何快速定位到文件某一行

准确快速地定位到文件某一行对本系统的实现至关重要，主要应用在如下场景：1）打开文件时，由于文件大小很大，不可能将整个数据文件全部加载，因此只能加载文件开头的若干行。这种情形下就要求快速取出文件的前若干行出来，涉及到文件的定位；2）用户使用鼠标向下滚动文件，当需要显示的文件内容超出了前若干行之后，需要动态地进行继续加载，同样需要文件的定位；3）用户想要直接跳转到文件的某一行查看相应内容。需要注意的是，此处的文件定位是粗粒度的，不能通过指定一个文件指针的偏移获得，因为文件每一行所占的字节数都可能是不等的。

Pandas 是 Python 的一个十分流行的数据分析第三方包。针对表格格式的文件，Pandas 提供了 *read_csv* 和 *read_table* 两个方法。这两个方法底层均使用 C 语言实现，上层提供了多种参数以实现定制化的数据读取。例如，指定 *read_csv* 方法中的 *skiprows* 参数，可使该方法跳过文件的前若干行开始读取；指定 *nrows* 参数，可以使该方法读取指定的行数；指定 *chunksize* 参数，可使该方法分块读取文件。这些参数使得用户在使用 Pandas 的 IO 方法读取大型文件时既便捷又高效。

本文根据上述的 Pandas IO 方法设计了一种“按需读取数据”的策略。该策略的具体解决方案如下：当用户尝试打开文件时，服务器预取并返回文件的前一千行；同时系统的前端通过 JS 绑定鼠标滚轮下翻事件，当用户将鼠标向下翻滚超过一定范围之后，页面向服务器请求接下来的数据；当用户使用“跳转至某一行”的功能时，服务器使用 Pandas 的 IO 接口进行相应行内容的读取。由于 Pandas IO 底层使用 C 语言来实现，可以对文件进行分块读取，因此跳转功能消耗的时间很少。图 2 为文件大小不同时，使用跳转功能跳转到文件正中间位置的性能情况，实验环境为五个 CPU 节点，64GB RAM 的物理集群，每个节点有一颗 Intel E5 2650 CPU，拥有 8 个物理核心。图表显示即使文件大小上升到 10GB，本项目所设计的跳转功能也能在 20 秒之内跳转到该文件中间的位置。

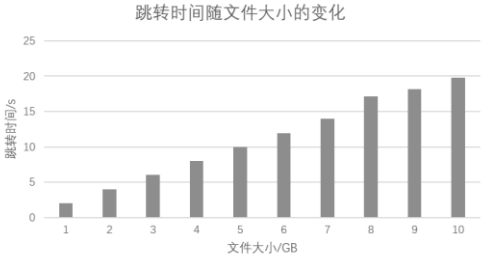

图 2 跳转时间随文件大小的变化

3.2 如何对数据进行并行处理

Docklet 的一个用户实例可以创建一个独有的虚拟集群，这使得 Docklet 在数据分析上有天然的优势——可以在虚拟集群上并行化处理数据，从而提高数据分析效率。本系统中利用 Docklet 的虚拟集群在后端进行表格数据的并行处理。用户在创建 Docklet 集群的每个节点时，通过指定相应类型的镜像来创建，并行处理的脚本文件与可执行文件即事先保存在镜像中。

数据并行处理的时机主要在对整个表格以某一列为键值进行排序、求最大值、求平均数、求和等操作时。

在上述的几个常用的统计操作中，求最大值、求和、求平均值属于中间结果较少的运算。例如使用 Map-Reduce 模型求解一列数的最大值时[16]，运算的中间结果占用内存空间大小随着数据的归并而降低，直到最后只剩下 1 个数。因此这样的统计操作占用的内存空间比较少，可直接用现有的多节点并行框架实现。本系统中设计了针对这几种统计操作的 Spark 算法，通过搭建在 Docklet 虚拟集群上的 Spark 集群来实现。以针

对文件的某一列求和为例，算法先使用 map 函数将表格文件某一列抽取出来，再使用 reduce 函数归并计算该列的和，伪代码如算法 1。

Algorithm 1: Parallel Sum

Input $filename; column$
Output sum
1. set up spark configuration
2. $textfile \leftarrow$ spark file of $'filename'$
3. $lines \leftarrow textfile.\text{flatMap}(x \rightarrow x.\text{split}('\backslash n'))$
4. $lines \leftarrow lines.\text{map}(x \rightarrow x.\text{split}(',')_{\text{column}})$
5. $sum \leftarrow lines.\text{reduce}(x, y \rightarrow x + y)$

算法 1 并行求和

但是排序操作就不同与上述几种，其中间结果占用内存很大，而且其内存占用不会随着时间的增长而降低，排序的结果和输入数据的大小是相同的[15]。另一方面，考虑到本系统中文件是以一千行为一个显示单位呈现给用户的，因此排序操作实际上简化为了求输入数据中的前一千大的数，因此这种情形下使用基于 Spark 不再高效。本系统采用了一种使用分治策略求前 k 大数的算法 DC-Top-k[13]，该算法先将数据切割成块（要求块数大于等于 k），在每块数据中求取一个局部最大值 *local_max* ，最后求出所有局部最大值中的最小值 *threshold*，然后遍历整个文件，找出所有比 *threshold* 大的数据，再对这些数据进行快速排序，最后得出排序结果中的前 k 大数。算法的伪代码如算法 2 所示。

DC-Top-k 算法使用分治的基本思想，因此非常适合并行实现，本系统中采用了其 MPI 并行化版本。并行版本的 DC-Top-k 总体分为五步进行。第一步，master 节点计算出需要排序的文件的行数，再按照行数和 k 计算出每个 worker 读取数据的起始文件偏移；第二步，每个 worker 节点从相应比例的偏移处开始读取文件数据，并计算出局部的 *threshold*，并将其发送给 master 节点；第三步，master 节点计算全局的 *threshold*，并分发给每个 worker；第四步，worker 找出本地数据中比 *threshold* 大的数据，并将这些数据发送给 master；第五步，master 汇总各 worker 发来的数据，进行快速排序，取前 k 大的数作为输出。

当前端发来一个对文件排序的请求时，后端即做出响应，返回该文件最大（或最小）的前 1000 行。

Algorithm 2: DC Top k algorithm

Input $r_0, \ldots, r_{n-1}; k$
Output $top_0, \ldots, top_{k-1}$
1. Divide $r_0, \ldots, r_{n-1}$ into k groups
2. **for** $i \leftarrow 0$ to $k-1$ **do**
3. $Max_i \leftarrow$ the maximum of group i
4. $Maxloc_i \leftarrow$ the position of Max_i in th group i
5. **endfor**
6. $threshold \leftarrow \min(Max_0, \ldots, Max_{k-1})$
7. $G_{min} \leftarrow$ the group id of $threshold$
8. $Q \leftarrow 0$
9. **for** $i \leftarrow 0$ to $k-1$ **do**
10. **if** $i = G_{min}$ **then**
11. **continue**
12. **endif**
13. **for** $j \leftarrow 0$ to $N-1$ **do**
14. **if** $r_{i*N+j} > threshold \,\&\, j \neq Maxloc_i$ **then**
15. $r_Q \leftarrow r_{i*N+j}$
16. $Q \leftarrow Q + 1$
17. **endif**
18. **endfor**
19. **endfor**
20. **for** $i \leftarrow 0$ to $k-1$ **do**
21. $r_{Q+1} \leftarrow Max_i$
22. **endfor**
23. $top_0, \ldots, top_{k-1} \leftarrow \max\_k(r_0 \sim r_{k+Q-1})$
24. **return** $top_0, \ldots, top_{k-1}$

算法 2 DC-Top-k 算法

3.3 如何进行有效的可视化

本系统中提供的可视化有两种：1）针对部分数据的绘图操作；2）能够显示数据整体特征的可视化功能。

（1）针对部分数据的绘图操作。

打开一个表格文件之后，用户可选中相应区域进行直方图、折线图或者饼图的绘制。系统前端在监听到相应的事件之后，将请求发送给后端，后端直接对数据进行处理，使用 python 第三方包 matplotlib[14]绘图，然后传回前端，在浏览器中显示。

（2）反应整体数据特征的可视化操作。

与（1）类似，反应数据整体的可视化操作同样不直接在前端使用 JS 执行。因为当数据量很大时，如果直接在前端通过 JS 进行可视化，浏览器的负载将非常大，使用户体验下降。因此本系统采取如下策略来实施对数据整体的描述：

用户点选表格文件的某一列，前端监听到该事件之后，将相应的请求发送到服务器，服务器对该文件相应的列进行数据统计，找出该列数据的极大值和极小值，并将该区间均分为若干段（默认为 1000），然后再统计该列数据落在各段内的频率，并将区间和频率键值对数据传回前端，前端再根据传回的数据绘制反应该列数据特征的热力图。前端热力图的绘制采用了前端可视化插件 Heatmap.js。算法伪代码如算法 3 所示。

Algorithm 3: Heatmap

Input $filename; column$
Output $heatmap$
1. front end send request to server
2. $column_{max} \leftarrow$ the largest number
3. $column_{min} \leftarrow$ the smallest number
4. k section $\leftarrow [column_{min}, column_{max}]$
5. **for** $section_i$ in $sections$ **do**
6. $\quad freq_i \leftarrow$ number of elements in this interval
7. **endfor**
8. send $freq_0, \ldots, freq_{k-1}$ to front end
9. front draw $heatmap$

算法 3 热力图

4 实例研究

本文在试验中生成了 1G 大小的随机数据文件，数据文件共有 10 列，对于每一列数据样本的总体 X 都有$X \sim N(500, 150^2)$，即文件每一列的总体都符合均值为 500，标准差为 150 的正态分布。试验环境为四节点的 Docklet 虚拟集群，每个节点有 4GB 内存，4 个虚拟 CPU 核心。对该文件进行打开、重排、跳转、选中绘图、全局可视化等操作试验，具体试验效果如下所示：

图 3 Jupyter Notebook 文件列表

4.1 表格显示效果

用户点击文件 1G.csv 之后，工具将该链接定向到相应的表格显示页面，表格显示效果如图 4 所示。

Skip To | Sort | Sum | Ave | Heatmap | Plot | 25000000 lines totally

	A	B	C	D	E	F	G	H	I	J
0	496	497	520	499	523	511	538	488	553	488.1
1	543	527	491	528	495	490	504	472	517	476
2	505	507	478	519	538	480	469	528	483	520
3	544	546	469	525	490	506	521	527	535	495
4	478	524	446	493	464	558	542	462	478	538
5	497	498	511	470	493	520	544	505	495	536
6	478	471	500	477	511	432	508	502	534	488
7	484	520	499	481	466	528	526	569	506	560
8	464	491	555	485	502	457	487	499	474	488
9	545	490	500	504	509	532	490	431	476	496
10	539	511	507	515	523	512	537	534	505	508
11	532	489	533	462	460	489	490	435	515	553
12	483	470	515	494	450	472	516	505	521	541
13	458	418	510	484	513	485	447	511	503	495
14	530	497	495	482	484	502	486	545	473	469
15	457	472	457	501	487	482	496	522	524	466
16	492	510	518	503	508	459	473	504	532	517

图 4 表格显示效果

4.2 排序效果

用户点击列标签 B 之后选中列区域，然后点击 Sort 按钮对所有数据按列 B 的值进行重排，重排之后效果如图 5 所示。

Skip To | Sort | Sum | Ave | Heatmap | Plot | 100000 lines totally

	A	B	C	D	E	F	G	H	I	J
0	512	342	526	470	494	462	464	493	466	483
1	501	357	527	470	474	465	497	514	469	525
2	514	362	496	524	511	481	509	492	468	430
3	529	370	484	481	504	463	462	484	491	521
4	523	374	543	551	479	563	518	515	493	491
5	464	377	531	527	501	544	466	471	437	521
6	533	377	478	516	524	492	494	471	467	489
7	505	379	497	517	469	524	475	493	515	506
8	523	381	460	515	485	497	498	475	502	470
9	478	383	484	436	507	533	522	474	501	519
10	512	384	533	514	515	492	506	556	461	462
11	482	386	443	501	498	518	521	522	464	504
12	511	386	510	470	507	522	525	521	487	466
13	505	387	490	559	487	516	407	501	496	548
14	509	388	502	501	527	458	526	575	509	497
15	518	389	542	472	458	526	442	524	525	487
16	502	389	499	463	491	491	495	470	460	543

图 5 排序后效果

4.3 求和效果

用户点击列标签 B 之后选中列区域，然后点击 Sum 按钮对列 B 进行求和运算，求和结果显示在第 0 行，效果如图 6 所示。

Skip To | Sort | Sum | Ave | Heatmap | Plot | 100000 lines totally

	A	B	C	D	E	F	G	H	I	J
0		49930792.0								
1	529	505	494	485	485.1	549	473	467	528	495
2	552	443	528	514	545	517	475	524	423	494
3	450	481	496	526	511	531	485	469	510	507
4	495	502	499	499	450	511	486	443	503	483
5	522	523	454	558	496	518	471	503	539	485
6	493	497	522	486	519	553	474	512	568	507
7	542	519	527	498	540	406	480	517	446	427
8	527	470	556	465	561	499	490	497	511	540
9	536	525	542	461	479	531	452	497	505	451
10	543	505	475	569	546	522	474	484	523	548
11	478	503	500	540	523	484	502	510	440	439
12	508	538	536	529	513	509	516	510	486	475
13	486	540	505	466	534	528	476	515	514	558
14	460	474	471	534	482	532	504	537	474	517
15	523	528	534	475	521	460	490	488	491	551
16	479	509	537	512	561	446	502	478	519	469

图 6 求和效果

4.4 热力图效果

用户点击列标签之后选中列区域，然后点击 Heatmap 按钮对文件的列 B 数据进行统计，得出

列 B 数据分布的热力图，效果如图 7。可以看出此即为正态分布的热力图。

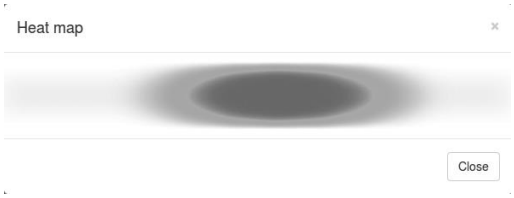

图 7 热力图效果

4.5 折线图绘制效果

用户选中一行数据，点击 Plot 下拉菜单中的 Line Chat 按钮进行折线图绘制，效果如图 8 所示。

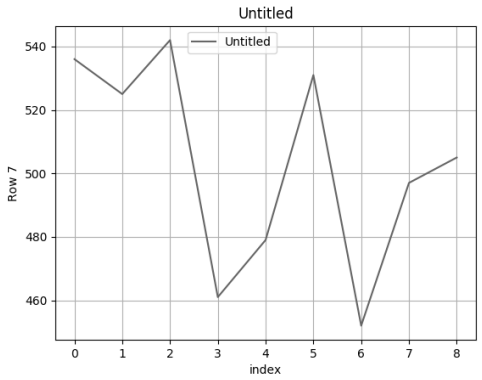

图 8 折线图效果

4.6 排序性能

以排序为例，本文将本项目使用的“排序”算法 DC_top_k 的性能与传统串行算法的性能做了对比，效果如图 9 所示。从加速比可以看出，当文件大小增大时，串行算法的劣势越为明显，而 DC_top_k 算法的优势也越为显著。当文件大小增长到 10GB 时，并行算法仍能在 30s 左右的时间内快速得出结果。

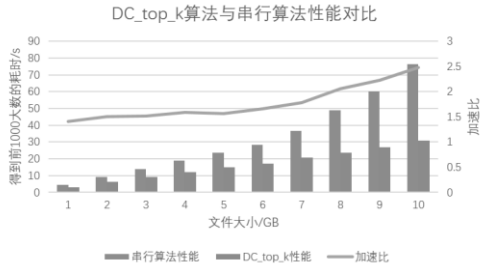

图 9 DC-Top-K 性能

结束语 本文分析了现有的数据分析和可视化工具的不足，创造性地将数据中心操作系统、数据分析、数据可视化以一种用户友好的方式结合起来，使用后端实现并行计算、前端实现可视化的思路，实现了一种基于 Web 的云端大数据可视化工具。用户使用本系统进行可视化操作，无需编程，只需简单的拖动和点击，即可对大型数据文件进行快速分析和可视化。本文的结尾通过实例研究，说明本文所提出方案具有可行性和有效性。

近些年来，数据可视化技术越来越成熟，图表类型越来越纷繁复杂。本文虽实现了一种云端大数据处理和可视化的解决方案，但是支持的图表类型尚不完善，数据操作的类型尚待进一步丰富。与此同时，如何让 Docklet 在用户创建虚拟集群的同时动化部署本工具，也是一个值得进一步研究的问题。

参考文献

[1] Bo An, Junming Ma, Donggang Cao, Gang Huang, Towards Efficient Resource Management in Virtual Clouds, The 8th International Workshop on Joint Cloud Computing (JCC2017), Atlanta, USA, Jun 5-8, 2017

[2] Donggang Cao, Bo An, Peichang Shi, Huaimin Wang, Providing Virtual Cloud for Special Purposes on Demand in JointCloud Computing Environment, JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY, 32(2):211-218, Mar 2017, DOI: 10.1007/s11390-017-1715-1

[3] Yujian Zhu, Junming Ma, Bo An, Donggang Cao, Monitoring and Billing of a Lightweight Cloud System Based on Linux Container, The 8th International Workshop on Joint Cloud Computing (JCC2017), Atlanta, USA, Jun 5-8, 2017

[4] Bo An, Xudong Shan, Zhicheng Cui, Chun Cao and Donggang Cao, Workspace as a Service: an Online Working Environment for Private Cloud, 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE), San Francisco, 2017, pp. 19-27. DOI:10.1109/SOSE.2017.11

[5] Kluyver, Thomas, et al. ″Jupyter Notebooks-a publishing format for

reproducible computational workflows." ELPUB. 2016.

[6] McKinney, Wes. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.", 2012.

[7] McKinney, Wes. "pandas: a foundational Python library for data analysis and statistics." Python for High Performance and Scientific Computing(2011): 1-9.

[8] Infogram https://www.infogram.com.

[9] Dory, Michael, Allison Parrish, and Brendan Berg. Introduction to Tornado: Modern Web Applications with Python. " O'Reilly Media, Inc.", 2012.

[10] Zaharia, Matei, et al. "Spark: Cluster computing with working sets." HotCloud 10.10-10 (2010): 95.

[11] Gropp, William, Ewing Lusk, and Rajeev Thakur. Using MPI-2: Advanced features of the message-passing interface. MIT press, 1999

[12] Handsontable https://www.handontable.com

[13] Zhengyuan Xue, Ruixuan Li, Heng Zhang, Xiwu Gu, Zhiyong Xu, "DC-Top-k: A Novel Top-k Selecting Algorithm and Its Parallelization" of the 45th International Conference on Parallel Processing, 2016, pp. 372-375

[14] Hunter, John D. "Matplotlib: A 2D graphics environment." Computing In Science & Engineering 9.3 (2007): 90-95.

[15] Vitter J S. External memory algorithms and data structures: Dealing with massive data[J]. ACM Computing surveys (CsUR), 2001, 33(2): 209-271.

[16] Yang H, Dasdan A, Hsiao R L, et al. Map-reduce-merge: simplified relational data processing on large clusters[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. ACM, 2007: 1029-1040.

[17] Team R J S. Requirejs[J].

[18] Handbook of data visualization[M]. Springer Science & Business Media, 2007.

李炎，北京大学信息科学技术学院本科生，专业方向为计算机科学与技术，联系电话 1880116807，邮箱 1400012951@pku.edu.cn

马俊明，北京大学软件与微电子学院博士生，专业为软件工程，联系电话 13521329745，邮箱 mjm520@pku.edu.cn

安博，北京大学信息科学技术学院博士生，专业为计算机软件与理论，联系电话 17710235556，邮箱 anbo@pku.edu.cn

曹东刚，北京大学信息科学技术学院副教授，研究方向为云计算，分布式计算等，联系电话 62757670，邮箱 caodg@pku.edu.cn