Understanding and Improving the Diagnostic Workflow of MapReduce Users *

Jason D. Campbell[τ], Arun B. Ganesan[†], Ben Gotow[†], Soila P. Kavulya[†], James Mulholland[†], Priya Narasimhan[†], Sriram Ramasubramanian[†], Mark Shuster[†], Jiaqi Tan[§]

Intel Labs Pittsburgh[τ] Carnegie Mellon University[†] DSO National Laboratories, Singapore[§]

ABSTRACT

New abstractions are simplifying the programming of large clusters, but diagnosis nontheless gets more and more challenging as cluster sizes grow: Debugging information increases *linearly* with cluster size, and the count of intercomponent relationships grows *quadratically*. Worse, the new abstractions which simplified programming can also obscure the relationships between high-level (application) and low-level (task/process/disk/CPU) information flows. In this paper we analyze the workflow of several users and systems administrators connected with a large academic cluster (based the popular Hadoop implementation of the MapReduce abstraction) and propose improvements to the diagnosis-relevant information displays. We also offer a preliminary analysis of the efficacy of the changes we propose that demonstrates a 40% reduction in the time taken to accomplish 5 representative diagnostic tasks as compared to the current system.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: User-centered design

General Terms

Human Factors

Keywords

user study, cloud-computing, diagnosis, MapReduce

1. INTRODUCTION

MapReduce [7] is a programming paradigm, proposed by Google, that enables scalable analysis of large datasets by distributing data processing across a cluster of nodes. MapReduce simplifies parallel programming using two abstractions namely: `maps` which breakdown the data-processing task into manageable chunks that run in parallel across the cluster, and `reduces` which consolidate the results generated by maps. MapReduce clusters can process large amounts of data–at Google alone, more than 100,000 MapReduce jobs process more than 20 PB of data daily [7]. An open-source implementation of MapReduce, known as Hadoop [18], is widely used for data processing at companies such as Yahoo! and Facebook, as well as for academic research [10].

Even though MapReduce makes developing parallel applications simpler, the complex interactions between components can cause errors to propagate across the system, obscuring the root-cause of the problem. In addition, poorly-written MapReduce applications can degrade the performance of other applications running concurrently on the cluster. These complex failure patterns, coupled with the sheer scale of clusters make it challenging to diagnose problems in a timely fashion.

To better understand the obstacles to diagnosis, we interviewed Hadoop users and system administrators of a large academic cluster. We had initially planned to focus our interface re-design efforts to support diagnostic workflows of system administrators. However, after conducting several interviews with system administrators, we discovered that they were already efficient at diagnosing infrastructural problems, *e.g.*, disk failures or network congestion, due to their years of experience. The administrators were less familiar with Hadoop so the task of troubleshooting MapReduce problems often fell to Hadoop users. We decided that the focus of an improved diagnostic interface should be to support Hadoop users so that they do not have to approach the system administrators every time they had a problem.

At present, the users diagnose problems by monitoring resource consumption, and browsing Hadoop logs using disparate web-based interfaces. We observed that users had to constantly switch between interfaces to perceive connections between the data and draw conclusions about the fault state of the system. Even with the

ACM CHIMIT '11, December 4, 2011, Boston, MA, USA.

combination of interfaces, certain information was not discernable and users had to resort to querying system state via the command-line interface of the underlying operating system.

This paper articulates the design principles for building a consolidated interface that improves the diagnostic workflow of Hadoop users by offering a "one-stop-shop" for visualizing the state of system. We illustrate these principles using an interface prototype, and perform a preliminary evaluation using keystroke-level modeling [4] to estimate the time taken to perform a set of tasks. We observed that the prototype achieved a 40% reduction in the time-taken to accomplish 5 diagnostics tasks when compared to the existing system.

2. ENVIRONMENT, USERS, AND TOOLS

Environment. Our study analyzed the diagnostic workflows of Hadoop users and system administrators of a 64-node cloud-computing cluster for data-intensive research at a university, which supports over 100 users. Hadoop [18] is an open-source implementation of Google's MapReduce [7] framework that enables distributed, data-intensive, parallel applications by decomposing a massive job into smaller (`Map` and `Reduce`) tasks and a massive data-set into smaller partitions, such that each task processes a different partition in parallel.

A Hadoop job consists of a group of `Map` and `Reduce` tasks performing some data-intensive computation. Hadoop automates job scheduling and allows multiple users to share the cluster. Users of the cluster run a diverse set of data-intensive workloads such as large-scale graph mining, text and web mining, natural language processing, machine translation problems, and data-intensive file system applications.

Users. During our study, we interviewed 3 Hadoop users, and 2 system administrators. The Hadoop users were researchers familiar with configuring Hadoop, writing and running MapReduce jobs, and analyzing the output generated by their jobs. The users had varying levels of experience with Hadoop: one of the users was a novice user, while the other two were advanced users. Hadoop users troubleshoot their MapReduce jobs when they suspect that the job has been running longer than usual, or when their jobs fail due to exceptional conditions such as insufficient file privileges. The root-causes of these problems range from bugs in their MapReduce jobs (*e.g.* infinite loops), configuration problems such as allocating insufficient memory to complete their jobs, to the occasional disk failure. As users share the cluster, they sometimes experience contention for resources with other jobs. The cluster currently does not support task preemption in which runaway tasks are killed to make room for other tasks in the system.

The system administrators were responsible for maintaining the overall health of the cluster by: (i) setting up hardware and software on the cluster; (ii) troubleshooting infrastructural problems (*e.g.*, hardware failures, misconfigurations); (iii) upgrading hardware and software; and (iv) maintaining user accounts. Both administrators we inteviewed had more than 5 years of experience managing clusters. The administrators were also less familiar with Hadoop, so the task of troubleshooting MapReduce problems was typically performed by Hadoop users.

The Hadoop users try to diagnose problems either on their own, by soliciting help from the cluster mailing list, or by escalating the problem to the system administrators. The users were interested in distinguishing between the following diagnostic scenarios:

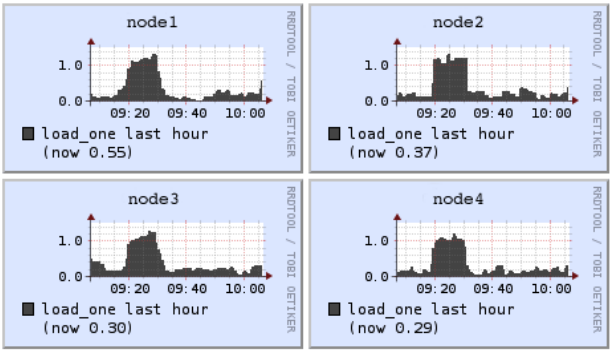
1. *Bugs in their MapReduce jobs:* Users sometimes make mistakes when developing their MapReduce jobs, such as referencing incorrect files, assigning insufficient file permissions, and software bugs such as infinite loops. Once they identify the bug, they can fix their code and re-run their jobs.

2. *Legitimately slow jobs:* Some tasks within their MapReduce jobs are legitimately slower than others because they are processing more data. Users can ignore these discrepancies in task durations, and allow their jobs to run to completion.

3. *Contention with other jobs:* As users share the cluster, sometimes buggy MapReduce jobs can degrade the performance of other jobs running in the cluster. For example, infinite loops in tasks or jobs which inadvertently fill up the temporary directories on nodes can interfere with other jobs in the system. Users can contact the owners of these buggy jobs or system administrators to resolve these issues.

4. *Infrastructural problems:* Users can escalate problems to the system administrators especially if they suspect an infrastructural problems, *e.g.*, disk failures, insufficient disk space or a cluster-wide misconfiguration.

Tools. Users primarily rely on three tools to diagnose problems in their jobs namely: (i) `Ganglia` (version 3.1.7) [9], a scalable distributed system monitor tool for high-performance computing systems such as clusters and grids that allows users to remotely view live or historical statistics (such as CPU load averages or network utilization as illustrated in Figure 1(a)); (ii) the Hadoop (version 0.20.1) web interface [18] that allows users to browse application logs and monitor the progress of the map and reduce tasks that constitute their jobs, as illustrated in Figure 1(b); and (iii) the



(a) Screenshot of Ganglia monitoring tool showing CPU utilization on several nodes over the past hour.

Hadoop job_201010180929_0001 on node1

User: user2
Job Name: random-writer
Job File: hdfs://node1:9000/grid0/hadoophome/0.18.3/mapred/system/job_201010180929_0001/job.xml
Status: Running
Started at: Mon Oct 18 09:30:17 EDT 2010
Running for: 9mins, 46sec

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	65.00%	20	0	7	13	0	0 / 8
reduce	0.00%	0	0	0	0	0	0 / 0

(b) Screenshot of Hadoop web interface showing progress of `Map` and `Reduce` tasks in a single job.

Figure 1: Users query disparate interfaces to track resource usage and progress of MapReduce jobs.

operating system terminal that facilitates more powerful exploration such as querying the real-time resource usage (*e.g.*, CPU utilization), of nodes and processes, pinging remote nodes to determine their status, `tail`-ing logs to track the real-time job progress, and dumping the process state of Hadoop jobs.

In addition, system administrators use Nagios [14], an open-source application that monitors nodes and services and alerts administrators when things go wrong. They also make use of a ticketing system that allows them to keep track of both open and resolved problems.

3. HUMAN-CENTERED DESIGN

We used a human-centered design methodology [3] to explore the diagnostic workflows of Hadoop users. Our aim was to identify common workflows in their day-to-day tasks, and identify breakdowns and opportunities for improving the existing diagnostic interfaces. Before conducting our research, we performed an affinity diagramming session and identified focii that provided direction for our user studies. During the diagramming session, we noted questions we had about the work process, attitudes, and needs of system administrators and Hadoop users.

We consolidated these notes into themes and identified two primary areas of inquiry: (i) *discovery*, the process in which Hadoop users realize a problem exists in the system; and (ii) *diagnostics*, the process in which administrators find the root cause of a problem. The diagnostics focus included questions such as *when are problems escalated to other administrators?* and *who is involved in a problem response?*

We recruited five participants for our study: 3 Hadoop users, and 2 system administrators. To understand the diagnostic workflows of these users, we conducted contextual inquiries [3]. Contextual inquiry is a human-centered research method in which researchers observe users performing their work in context and interrupt briefly to ask questions as they arise. We believe that *what the user says he does, and what he actually does can be very different things.* By observing users and their work flows, we gathered data that informed a design process supporting their everyday tasks and goals.

We initially expected that system administrators would be the primary target of our research. However, after conducting two contextual inquiry sessions, we decided to focus on Hadoop users rather than system administrators. Hadoop users were less aware of the overall state of the system, had fewer access privileges, and were more confused by problems when they arose. The challenges that Hadoop users faced provided opportunities to develop tools to improve their workflow efficiency.

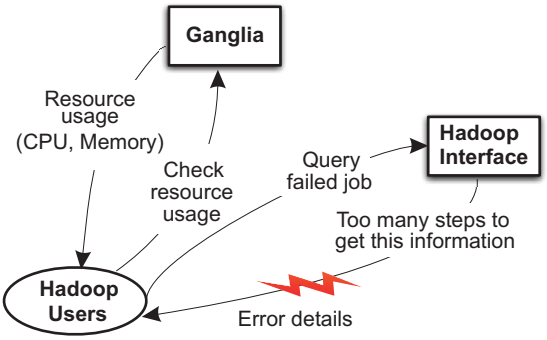
4. USE CASES

Through our contextual inquiries, we identified common workflows of Hadoop users when diagnosing problems in their jobs. In what follows, we describe cases that illustrate important aspects of the diagnostic workflow, focusing on breakdowns because these are areas that require the most immediate attention.

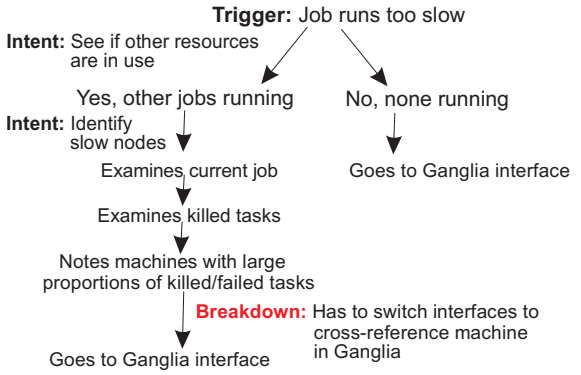
4.1 Case 1: Misleading application bug

Bob[1] begins a Hadoop job from the terminal. He monitors the job progress from the terminal window. *"I like to monitor progress from the terminal. If there is a problem, I can see it faster."* After receiving sporadic feedback about job progress, Bob switches windows to the Hadoop web interface and refreshes the page. *"I think I'm competing with another job".* He sees that another job is being run by a different user and restarts his job to test his theory. To do this, he must scroll through the job output on his terminal to find the kill command, then copy and paste the command in the terminal. *"If the job is very long, you have to go back several screens to get it."*

[1]Not his real name.



(a) Flow model showing how users interact Hadoop and Ganglia. The model highlights that users take too many steps to query error detail for failed jobs.



(b) Sequence diagram showing the steps taken to diagnose slow Hadoop jobs. Users often switch between multiple interfaces to accomplish their goals.

Figure 2: Examples of models generated during the contextual design process.

After modifying his code, Bob attempts the job again, but this time the job fails. The terminal tells him that the job failed, but it does not contain sufficient information to diagnose the problem. Bob returns to the web interface to gather more information and refreshes the job page again. He determines there is a problem by noticing that the number of failed tasks on his job is greater than zero. *"If this number is non-zero, basically there is something wrong with your program. If the number is low, two or three, the node may be unstable."* He then clicks on the number of failed tasks in the web interface to view the error messages, but he must navigate to another page to access a full error log. *"It's kind of painful to do this"*. From this he determines the cause to be due to a syntax error in his code.

4.2 Case 2: Overloaded node

Jeff[2] retrieves an email message posted to the cluster mailing list requesting assistance with troubleshooting a job that is failing. Jeff examines the failed job in the Hadoop web interface. He selects the job from the job list and subsequently clicks on the number of killed tasks from the job summary page. He observes the list of killed tasks and identifies the node on which the task was killed, *"ahh, this was the machine"*, which is labeled as node 13, *"sometimes it's not that obvious"*. Jeff changes browser tabs to the Ganglia web interface where the individual cluster nodes are monitored. He scrolls through the list of nodes to find node 13, which he identified as being troublesome in the Hadoop interface and selects it. He explains, *"Sometimes it's not just your job that caused the problem, sometimes someone else might cause you a problem"*. The resulting page displays hourly plots of all the metrics collected from node 13. He glances at the first few graphs, *"these are more likely to be important"*, he quickly scrolls the rest of the page, *"this is too much detail, there is too much data to look at"*. Back at the summarized node 13 graphs, he synthesizes the information from multiple graphs to determine that the node was under heavy load during the period over which his task ran.

[2]Not his real name.

5. IDENTIFYING WORKFLOW GAPS

We consolidated the interview transcripts from the five contextual inquiries, and built models in accordance with the contextual design process namely: flow, sequence, cultural, artifact, and physical models. These five models expose common themes in the Hadoop users' environment and workflows. We present brief descriptions of these models, and their significance in exposing aspects of Hadoop users' workflows.

Flow models. Flow models help us to understand the roles of different users, detailing the interactions between them while performing their duties. The flow models motivated the need for a consolidated diagnostic interface that displayed both the Hadoop job information and the associated resource consumption, since users often used these tools in conjunction with each other when diagnosing problems (see Figure 2(a)).

Sequence models. Sequence models provide insight into the steps users take to perform each of their tasks. These models help researchers to understand what triggers a user to perform a particular step and exposes the breakdowns the user faces while trying to accomplish their goals. For example, the sequence models revealed that Hadoop users must often switch between multiple interfaces such as Hadoop, Ganglia and a terminal window to accomplish their goals (see Figure 2(b)).

Cultural models. Cultural models identify the social and behavioral influences that affect users' actions. Our consolidated cultural model revealed that system administrators were already efficient because of their familiarity with the workflow and years of experience. The models also revealed an interesting mismatch between the expectations of system administrators and users–Hadoop users thought that more issues should be escalated to administrators, whereas administrators thought that many issues could be resolved without their help. We decided that the focus of an improved diagnostic interface should be to support Hadoop users so that they do not have to approach the system administrators every time they had a problem.

Artifact models. Artifact models expose the role of physical tools, documents, and other artifacts in the day-to-day work of the user. In our investigation of Hadoop users and system administrators, the prominent artifacts were the web browser, and the terminal window.

Physical models. Physical models highlight properties of the physical environment that impact the user's ability to accomplish their tasks. These models helped us analyze screenshots from the existing diagnostic interface to identify the features and functionality employed by Hadoop users. They revealed that many important displays of data were hidden or took multiple steps to reach. For example, viewing the data associated with a task required copying the task ID, switching to another view several clicks away, and then pasting the task ID.

5.1 Gaps in existing interfaces

The contextual inquiries revealed several important breakdowns in existing diagnostic interfaces used by Hadoop users, namely:

No single point for diagnosis-relevant information. We observed that users spent an inordinate amount of time locating key pieces of data spread across multiple systems. Users often juggled between multiple browser tab views to obtain the desired information.

Lack of information prioritization. In many cases, important data needed by the user is not displayed prominently and conveniently. The user often must go through several levels of navigation to find needed information.

Users unaware of the computational cost of their jobs. The Hadoop users in our cluster had no way of knowing how effectively they were managing resources on the Hadoop cluster. They were unable to assess their usage of the cluster, and were not motivated to limit their use of system resources in a way that would be beneficial to everyone in the user community. Displaying a visualization of resource consumption that indicates how the user is doing compared to the rest of the community would make users aware of their role in a larger context, and allow users to schedule their jobs more considerately.

Information overload. Another major breakdown we observed was the copious amount of information displayed to the user due to the large number of nodes in the cluster, and the approximately hundred different performance metrics collected from each node. Users were presented with stacks of graphs that commingled meaningful information with irrelevant data. For example, we observed that users were sometimes interested in viewing real-time statistics about resource consumption on a node. The real-time information is available as an annotation beneath each graph in the Ganglia interface. However, due to the sheer number of graphs displayed for each node, we found that users would bypass Ganglia, open a terminal window, and monitor the real-time resource via the `top` command. The `top` command succintly displays real-time resource consumption for processes running on a node.

6. DESIGN RECOMMENDATIONS

We propose five key themes to guide the design of an improved interface that supports the diagnostic workflow of Hadoop users. These themes were drawn from our observations of the major pain-points experienced by Hadoop users when diagnosing problems. The existing Hadoop and Ganglia interfaces have enough information for users to effectively diagnose problems–however, these interfaces are limited because certain metrics are not displayed well. The key design themes are: (i) personalize interfaces; (ii) prioritize information display; (iii) consolidate information; (iv) support data exploration; and (v) illustrate communal context.

6.1 Personalize interfaces

Different users need access to different types of information based on their level of experience, and their role within the system. Hadoop users are primarily concerned with making sure their individual jobs are running as expected, while system administrators are concerned with monitoring the overall health of the entire cluster. Support for personalized interfaces would ensure that the user's individual needs are met.

6.2 Prioritize information display

The data displayed on the interfaces must distill the most crucial information as quickly and as easily as possible. Users often consulted an interface looking for specific information, or intending to scan particular visualizations to detect anomalies. In both scenarios, the most important information should be displayed promi-

nently to allow users to quickly and accurately troubleshoot problems in their system. Organizing information using a hierarchy that fits the users' notion of priority can improve the users' workflows. For example, users had to click through several pages in the Hadoop web interface to access the error logs showing why their job failed. Displaying a snippet of the most recent exceptions in the logs alongside the failed job would ease the troubleshooting process. Another critical piece of information requested by Hadoop users was an indication of how long their current job had been running, and an estimate of how much more time was needed before it completed.

6.3 Consolidate information

The current interfaces lacked a single point for accessing diagnosis-relevant information. The Hadoop web interface provided information about a user's jobs and tasks, while Ganglia provided aggregate information about resource consumption at both the node and cluster level. Users constantly switched between interfaces to identify connections between the data. Even with a combination of interfaces, certain levels of information are not discernable. For example, users could not view the progress of tasks running on a particular node, and the associated resource consumption on the same interface.

6.4 Support data exploration

We observed that Hadoop users coped with the copious amounts of information available in the system by adopting top-down strategies for diagnosis. For example, a user would first identify the job that failed before zeroing in on the tasks that failed. If a user suspects that the problem is due to resource contention, the user would identify the nodes that the tasks were running on and look up their resource consumption on Ganglia.

Diagnostic interfaces should support data exploration strategies that allow users to form, and confirm their hypotheses on the root-cause of the problem. Developing dynamic and interactive visualizations, which allow clicking and hovering, would facilitate greater exploration. In addition, hyperlinks would allow users dynamic access to related information without entering a new query, or manually launching a new interface. Support for sorting, filtering and searching would allow users to find information (e.g., nodes or tasks) easily.

6.5 Illustrate communal context

The existing interfaces do not provide Hadoop users with enough information to understand the impact of their jobs on the cluster. For example, an indication that the current user's job is consuming 60% of cluster-wide memory, or that a user's task is running concurrently with 3 other tasks from unrelated jobs could help users distinguish between problems due to their jobs, and problems due to contention with other jobs. This would be particularly helpful in non-virtualized environments where resource isolation is not strictly enforced.

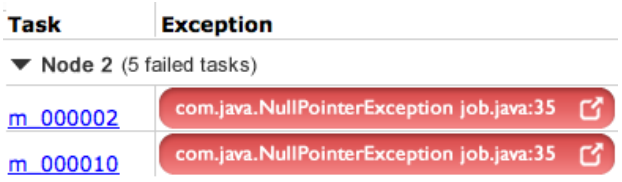


Figure 3: The prototype allows users to view all failed tasks for particular job, grouped by node.

7. INTERFACE PROTOTYPE

We developed a prototype of a diagnostic interface that incorporates 4 of the 5 key design themes identified by our consolidated models. The prototype does not address how to visualize the impact of a user's job on the community, but rather focuses on testing our recommendations for improving how users identify problems in their individual jobs.

We iteratively refined the design of the prototype using wireframes of the interfaces, and storyboards that helped us uncover flaws in our proposed designs. We implemented the interface prototype as a set of clickable screens using CogTool [5]. CogTool is an interface prototyping and evaluation tool that allows user interface designers to compare prototypes by predicting how long it will take a user to complete tasks in each prototype.

We describe how we incorporated 4 key design themes into our prototype through: (i) personalization; (ii) more prominent error notification, and data visualizations that prioritize information display; (iii) consolidation of job information and associated resource consumption in a single interface; and (iv) support for data exploration through search and filtering.

7.1 Personalization

The prototype personalizes information display by prioritizing the information displayed based on their username. Upon accessing the home page, the user is immediately presented with any errors that occurred on jobs that ran under their username. Each of the job categories (*i.e*, running, completed and failed) are sorted to display the user's jobs first so that information relevant to them is not obscured by jobs from other users.

7.2 Prominent error notification

We observed that users had to click through multiple screens to access error logs that they were interested in. The prototype prominently highlights failed jobs on the user's home page along with a snippet of the most recent task exception thrown prior to the job failure.

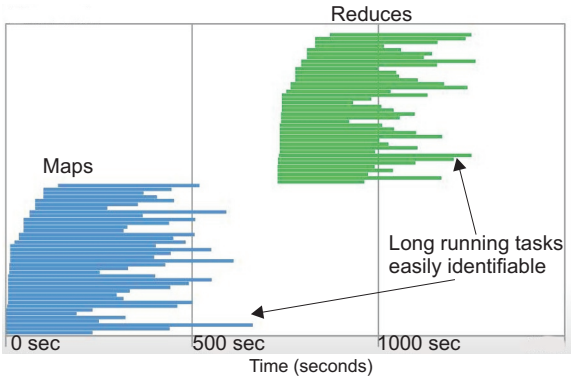


Figure 4: Swimlane graph charting the start and end times, and durations of `Map` and `Reduce` tasks for a single job. The graph also highlights the inherent structure of MapReduce jobs with map tasks completing before reduce tasks.

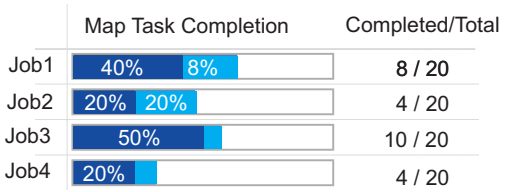


Figure 5: Progress bars display the overall proportion of tasks that are completed (dark blue) and currently running (light blue) for each job.

We display the time of the job failed, as well as the node on which the most recent task failed. In addition, we provide a link that allows users to view all failed tasks for a particular job, grouped by node. A snippet of the exception is displayed next to each failed task and is highlighted in red as shown in Figure 3. Users access the detailed error log containing the exception by clicking the error notification.

7.3 Improved data visualization

The current Hadoop web interface does not adequately visualize the large quantities of task data associated with each job. We observed that the users in our study ignored the visualizations because they did not capture the information they were looking for. Instead, users opted to glean information on task progress from the text descriptions available via the interface. We incorporated new visualizations into the prototype that succintly displayed the information requested by users namely: swimlane graphs, progress bars, and sparklines.

Swimlane graphs. Swimlane graphs [17] provide a more comprehensive representation of the performance of individual tasks in a job by charting the start and end times of individual tasks, while also receiving instantaneous feedback about task duration (see Figure 7.2). Users in our study indicated that determining the root cause of individual poorly performing tasks was the most elusive problem they encountered. Swimlanes allow users to quickly identify long-running tasks that might be contributing to the performance slowdown. In a functional implementation of our prototype, these lanes would be interactive allowing users to navigate to the task-specific page when a swimlane is selected.

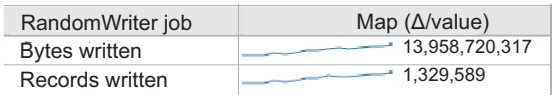


Figure 6: Sparklines showing the amount of data read by `Map` tasks over time next to each job.

Progress bars. We used a tiered, horizontal progress bar to visualize the overall proportion of tasks that are completed, and that are currently still running, for each job in the cluster (see Figure 5). The Hadoop web interface only displayed completed tasks.

Sparklines. Users expressed a need for visualizations that tracked historical trends in job performance, which was lacking in the Hadoop web interface. We addressed this need by visualizing the progress indicators, *e.g.*, the amount of data read by `Map` tasks over time using sparklines next to each job as shown in Figure 6. Sparklines [19] are small, high resolution graphics embedded in a context of words, numbers, images. Users can detect whether a job is experiencing poor performance by looking for flat lines or stutters in the sparkline.

7.4 Consolidating disparate interfaces

The prototype incorporates information from the existing diagnostic interfaces available to the user into a common workflow that eliminates the need for multiple disparate interfaces. Cluster-wide resource consumption, *e.g.*, CPU and memory utilization, traditionally displayed using Ganglia interface are now prominently displayed alongside information on job progress.

We observed that users were highly dependent on the terminal window to initiate and kill jobs, to navigate to specific logs, and to read error messages. However, the workflow for extracting job information from the Hadoop web interface for use in the terminal window was tedious– often involving copying and pasting of key information such as job IDs. The prototype addresses this by providing interactive access to common terminal tasks such as initiating and killing jobs.

7.5 Filtering and sorting data

The prototype allows users to explore the copious amounts of data available in the cluster by enabling

Table 1: Tasks performed during think alouds.

No.	Task
1	Locate the 6th line of the error log of a task for the job that failed
2	Kill the job that you were running previously
3	Determine the duration of the longest-running map task associated with your job
4	Determine the load average of the node that has the longest running map task
5	Identify the 5th data record read by the longest running map task

users to sort columns of data in-page, as well as filter by key term using a contextual search field. We organized data using tabs that grouped tasks based on whether they were completed, running, or failed. The prototype also features a simple checkbox to group tasks by node. This allows for users to quickly see if long-running or problematic tasks are endemic to a specific machine during troubleshooting. A search field is also available on job and task lists to allow for the quick filtering of specific users or task ranges on tables that can grow to span many pages. The prototype allows users to quickly sort tasks by duration to determine the longest running tasks, whose identification could serve as a starting point for troubleshooting performance problems.

8. EVALUATION OF PROTOTYPE

To evaluate the effectiveness of our interface redesign, we performed Think Aloud Usability Analysis [3], a human-centered design approach also known as think alouds. Think alouds are particularly effective in understanding breakdowns in interfaces, content and user flow. During think alouds, the user is given a predefined scenario and a set of tasks to complete. They are then asked to speak their thought process as they perform each of these tasks. Observing the user in this process produces valuable data about which tasks the user finds difficult to accomplish, or fails to accomplish and what are the problematic parts that caused this.

As part of our interface assessment, we also performed keystroke-level modeling [4] using the CogTool [5] interface prototyping and evaluation framework. Keystroke-level modeling is a method of estimating the time taken to perform each of the input tasks (usually through keyboard and mouse), and identifying inefficiencies in the sequence of tasks. Keystroke-level modeling also enables comparison of task durations across two different interfaces.

We conducted 3 think-aloud usability tests on the prototype interface with 3 different Hadoop users. Users were asked to perform the 5 tasks listed in Table 1. Our preliminary evaluation showed that our interface prototype achieved a 40% reduction in the time-taken to accomplish the 5 tasks in comparison to the existing interfaces.

8.1 Usability aspect reports

We analyzed screen capture and audio recordings from the think alouds, and generated usability aspect reports for each user, listing breakdowns and assigning severity ratings to each of the breakdowns. We consolidated the usability-aspect reports and used them as a guide to refine the prototype. The findings from the usability aspect reports are:

Need for appropriate sorting and filtering of tables. We found sorting and filtering to be a useful and familiar function for think-aloud participants. However, we did encounter some breakdowns when participants attempted to use sorting and filtering. Sorting and filtering were not always enabled for the columns that the user desired or expected. We found that our initial prototype did not clearly indicate which columns were sortable and which were not. Future iterations of the interface will clearly highlight filterable columns.

Error notifications need more context and less ambiguity. The design of the error notifications to display the stack trace when a notification is clicked was successful. However, some users were confused by what clicking on the error notification would display. Changing the error label to something more meaningful may help clear up this confusion. Users mentioned that error notifications were useful data, rightly positioned at the beginning of the navigation, and well-paired with the associated jobs or tasks. They also felt that it would be even more valuable to display larger snippets of error logs, along with recommended troubleshooting tips.

Provide instructions that explain unfamiliar visualizations. Swimlane visualizations provided a comprehensive representation of task durations. However, some users were not familiar with their use and took some time to figure out what the data meant. However, once familiar, they were able to quickly determine relative task lengths, identify short and long-running tasks, and identify how many tasks they had run. Providing instruction, or context upon interacting with the graph may make future revisions more successful in utilizing swimlanes visualizations.

8.2 Implementation recommendations

In addition to addressing the breakdowns in the prototype identified by the usability aspect reports, we propose the following recommendations when implementing the ful-fledged diagnostic interface.

Enable multiple access points to data. Our prototype provides access to node-level information via the task pages. This was a design decision based on user studies which informed us that Hadoop users rarely start off in

diagnosis by looking at node-level information. However node-level information is more useful for system administrators who are concerned with overall cluster health. A future prototype should contain provisions for accessing the node-level information through other navigation paths.

Re-configurable widgets. Our prototype uses a widget-styled layout for its web pages. We found that the way each user looks for information is different. Since the layout contains widget-styled blocks of data, the interface can be easily modified to provide a modular, reconfigurable experience, so that each user can have customized pages based on their needs.

Platform for communication. We observed that there was no support within the current diagnostic interfaces for users to consult each other about troubleshooting problems. Moreover the system administrators were heavily in need of communication to check if long-running jobs started by other users can be safely killed. They currently use email for this communication and it usually involves a long response time. An internal messaging system helping Hadoop users to communicate with other users in the same cluster would enable greater understanding of the health of the system, and of the jobs that others are running.

9. RELATED WORK

A number of tools and techniques for system administrators have used visualizations to present system information to help them manage their systems. Some of these visualizations specifically target request flows in distributed systems and applications. Magpie [1], X-trace [8], and Dapper [15] are techniques for tracing causal request paths in distributed systems, but they also offer support for visualizing requests whose causal structure or duration are anomalous to aid in diagnosis. These tools are designed for general distributed systems. In contrast, Mochi [17] is a Hadoop-specific log-analysis tool. It correlates Hadoop's behavior in space, time and volume, and extracts a causal, unified control- and data-flow model of behavior across the nodes of a cluster. The above visualizations can be incorporated in our designs, and in particular, we have used the Swimlanes visualization of Hadoop job behaviour from [17] in our design. SCUBA [11] is an interactive focus and context visualization framework for diagnosis in metro-mesh wireless health networks. While these tools have presented specific visualization techniques for various distributed systems, in this work, we use a human factors approach to study the challenges facing users of MapReduce systems, and present tailored visualizations and interfaces for helping end-users better use and manage their MapReduce jobs.

Some work has also used human factors approaches to study and design visualization for systems management. LiveRAC [13] is one such tool for browsing and correlating time-series data in large-scale systems. The authors built LiveRAC using a staged development process which incorporated interviews to solicit requirements, which they refined using various stages of prototypes. LiveRAC's main target users were Life Cycle Engineers (LCE), who were highly technical staff managing systems for of customers, whereas our target users were Hadoop end-users who are not system administrators, but due to the large scale of MapReduce clusters they need to understand the mechanics of the system to use the clusters effectively.

A secondary contribution of LiveRAC in in the use of a reorderable matrix of charts, with semantic zooming adapting each chart's visual representation to the available space. In addition, Ganglia [9] and Nagios [14] also use a reorderable matrix to display time-series data averaged over the past hour, day, week, month, or year. Ganglia web front-end caters to system administrators and users. These elements of interface design can all be incorporated in our designs to address the relevant challenges facing MapReduce users.

A number of commercial tools also visualize error logs, as compared to presenting regular system behaviour in the above techniques. Splunk [16] is a commercial tool that searches, monitors and analyzes log data in real time. It can also index data from multiple sources (logs, config data, `ps`, `systat`, `top`), and search or query data for specific strings. Artemis [6] provides a pluggable framework for distributed log collection, data analysis, and visualization. In addition, NetClinic [12] visualizes data from computer networks using directed graphs integrated with a multi-level automated analytic reasoning engine. However, these tools are targeted at system administrators rather than regular users, as we have targeted.

[2] conducted field studies observing and interviewing system adminstrators and presented 4 case studies. The sysadmins interviewed were seasoned professionals, and the systems that these sysadmins managed are likely to be on the scale of the MapReduce clusters our interviewed users work with. However, our targeted MapReduce end-users are likely to possess less technical depth than the sysadmins interviewed in [2], and nodes in a MapReduce cluster are likely to be more tightly coupled than in other typical server environments, increasing the complexity of understanding MapReduce system behavior for our users. [20] presented a method for empirically determining the user satisfaction of sysadmins of new tools which are deployed for for their use. They also identified four features of sysadmin tools that they would find helpful: accuracy of data, verification, reliability and credibility. These features are complemen-

tary to our findings, although our findings are specific to end-users of MapReduce clusters, albeit whose requirements of a diagnosis/management tool would be similar to those for sysadmins.

10. CONCLUSION

In this work, we have utilized a human-centered design methodology to develop an improved user-interface for end-users of the Hadoop MapReduce system to enable quick and effective diagnosis of performance problems of user jobs on MapReduce clusters. Through our user interviews and contextual inquiries, we found that although end-users of Hadoop MapReduce clusters are not system administrators, they often require system-level information, which is typically only available to system administrators, to help determine the source of a problem in their MapReduce jobs, and that users are confronted with large amounts of information due to the potentially large scales of MapReduce clusters. In this work, we have also made five design recommendations: (i) personalized interfaces; (ii) consolidated information; (iii) prioritized information display; (iv) support for data exploration; and (v) illustrating the communal context of each user's jobs. We implemented a prototype of the diagnostic interface, and evaluated the prototype using Keystroke-level modeling while in use by users, and using Think Aloud Usability Analysis. We found that our prototype interface was able to reduce the time taken for 3 common diagnostic tasks which we identified by 40%.

11. REFERENCES

[1] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *USENIX Symposium on Operating Systems Design and Implementation*, pages 259–272, San Francisco, CA, Dec 2004.

[2] R. Barrett, E. Kanodogan, P. Maglio, E. Haber, L. Takayama, and M. Prabaker. Field studies of computer system administrators: Analysis of system management tools and practices. In *CSCW*, Chicago, IL, Nov 2004.

[3] H. Beyer and K. Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann, 1st edition, September 1997.

[4] S. K. Card, T. P. Moran, and A. Newell. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM*, 23:396–410, July 1980.

[5] Carnegie Mellon University. Cogtool, 2011. `http://cogtool.hcii.cs.cmu.edu/`.

[6] G. F. Cretu-Ciocarlie, M. Budiu, and M. Goldszmidt. Hunting for problems with artemis. In *USENIX Workshop on Analysis of System Logs*, San Diego, CA, December 2008.

[7] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, 51:107–113, 2008.

[8] R. Fonseca, G. Porter, R. Katz, S. Shenker, and I. Stoica. X-Trace: A pervasive network tracing framework. In *USENIX Symposium on Networked Systems Design and Implementation*, Cambridge, MA, Apr 2007.

[9] Ganglia. Ganglia monitoring system, 2007. `http://ganglia.info`.

[10] Hadoop. Powered by Hadoop. `http://wiki.apache.org/hadoop/PoweredBy`.

[11] A. P. Jardosh, P. Suwannatat, T. Höllerer, E. M. Belding, and K. C. Almeroth. Scuba: Focus and context for real-time mesh network health diagnosis. In *Conference on Passive and Active Network Measurement,PAM*, pages 162–171, Cleveland, OH, April 2008.

[12] Z. Liu, B. Lee, S. Kandula, and R. Mahajan. Netclinic: Interactive visualization to enhance automated fault diagnosis in enterprise networks. In *IEEE Conference on Visual Analytics Science and Technology*, pages 131–138, Salt Lake City, UT, October 2010.

[13] P. McLachlan, T. Munzner, E. Koutsofios, and S. C. North. LiveRAC: interactive visual exploration of system management time-series data. In *Conference on Human Factors in Computing Systems, CHI*, pages 1483–1492, Florence, Italy, April 2008.

[14] Nagios Enterprises LLC. Nagios, 2008. `http://www.nagios.org`.

[15] B. H. Sigelman, L. A. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhagy. Dapper, a large-scale distributed systems tracing infrastructure. Technical Report dapper-2010-1, Google, April 2010.

[16] Splunk Inc. Splunk: The IT search company, 2005. `http://www.splunk.com`.

[17] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan. Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, San Diego, CA, June 2009.

[18] The Apache Software Foundation. Hadoop, 2007. `http://hadoop.apache.org/core`.

[19] E. R. Tufte. *Beautiful Evidence*. Graphics Pr, 1st edition, July 2006.

[20] N. Velasquez, S. Weisband, and A. Durcikova. Designing tools for system administrators: An empirical test of the integrated user satisfaction model. In *Large Installation System Administration (LISA)*, Nov 2008.