Timely Long Tail Identification Through Agent Based Monitoring and Analytics

Peter Garraghan, Xue Ouyang, Paul Townend, Jie Xu
School of Computing
University of Leeds
Leeds, UK
{p.m.garraghan, scxo, p.m.townend, j.xu} @ leeds.ac.uk

Abstract—The increasing complexity and scale of distributed systems has resulted in the manifestation of emergent behavior which substantially affects overall system performance. A significant emergent property is that of the "Long Tail", whereby a small proportion of task stragglers significantly impact job execution completion times. To mitigate such behavior, straggling tasks occurring within the system need to be accurately identified in a timely manner. However, current approaches focus on mitigation rather than identification, which typically identify stragglers too late in the execution lifecycle. This paper presents a method and tool to identify Long Tail behavior within distributed systems in a timely manner, through a combination of online and offline analytics. This is achieved through historical analysis to profile and model task execution patterns, which then inform online analytic agents that monitor task execution at runtime. Furthermore, we provide an empirical analysis of two large-scale production Cloud datacenters that demonstrate the challenge of data skew within modern distributed systems; this analysis shows that approximately 5% of task stragglers caused by data skew impact 50% of the total jobs for batch processes. Our results demonstrate that our approach is capable of identifying task stragglers less than 11% into their execution lifecycle with 98% accuracy, signifying significant improvement over current state-of-the-art practice and enables far more effective mitigation strategies in large-scale distributed systems worldwide.

Keywords—Long Tail, Stragglers, Distributed Systems, Data analysis, agent based, datacenter, Cloud computing

I. INTRODUCTION

Modern day services typically leverage interconnected distributed systems globally in order to fulfil consumer Quality of Service (QoS) demands and business objectives. This is particularly true when provisioning applications within the Cloud computing and Big Data domain - applications which require large amounts of computing power and storage capacity in order to process large quantities of data in an acceptable time frame. With the scale of systems increasing in both physical size and complexity, providers responsible for provisioning services are facing increasing challenges in mitigating - in a timely manner - the effect of emergent system behavior at scale which substantially impacts service QoS.

One such behavior is the Long Tail. This phenomena occurs when a distributed job - composed of multiple smaller tasks executing in parallel - incurs significant delays in completion due to a small subset of its parallelized tasks performing much slower than the other tasks within the same job known as stragglers, thus impeding job completion [2]. Research by both academia and industry has shown that the Long Tail problem imposes a significant challenge in providing timely and predictable application completion times, which is becoming increasingly difficult as systems increase in scale and complexity.

There have been a number of recent works that attempt to mitigate the effect of the Long Tail at runtime. These approaches primarily use speculative execution methods that leverage redundant computation [12][14], network congestion [9], and data locality [13]. While such works have been demonstrated to reduce the impact of the Long Tail, a universal challenge is the ability to identify potential straggler behavior as quickly as possible in order to mitigate Long Tail effects without breaching QoS time constraints. A common assumption is that all stragglers can be accurately identified within the system; in practice, this is challenging when considering the different task computation patterns within a system, as well as the requirement to identify stragglers in the shortest time frame. Furthermore, it has been identified in [16] that historical data of task execution can be leveraged as an effective means to calculate task progress, to allow for speculative task execution, and to avoid faulty nodes [17]. However, this work is not designed to identify and mitigate task straggler occurrence within system.

From studying both straggler mitigation mechanisms and offline analysis of task execution, there is a clear gap in the literature when it comes to combining both online and offline analytics of task execution together to identify straggler behavior within the least amount of time.

This paper proposes a model and implementation for automated analytic-driven Long Tail identification in distributed systems in order to enhance state-of-the-art Long Tail mitigation techniques. Specifically, our approach combines historical data analytics (to model task execution for different applications) with online agent-based monitoring and analytics of executing tasks (to identify stragglers caused by data skew). There are two core contributions of this work:

- A timely straggler identification mechanism leveraging both historical analysis and online agent based monitoring/ analytics to identify straggler tasks in a much reduced timeframe. Results demonstrate that our approach identifies stragglers less than 11% into a tasks lifecycle with a false positive rate of 1.59%.
- Empirical analysis of Long Tail Phenomena in two unique real-world production Cloud systems consisting of thousands of heterogeneous servers, demonstrating for the first time the prevalence and impact of stragglers due to data skew and faulty nodes. Our findings demonstrate that less than 5% and 3% of task stragglers causes 35% and 59% of total batch jobs within each system, respectively to experience Long Tail phenomena.

The paper is structured as follows: Section 2 discusses the background to the research; Section 3 presents related work;

Section 4 presents an empirical analysis of straggler impact in production Cloud datacenters; Section 5 presents the system model; Section 6 presents the experiments and evaluation of the proposed mechanism. Finally, Section 7 discusses conclusions and future work.

II. BACKGROUND

Big data analytics frameworks such as MapReduce, Dryad, Hadoop, and Spark decompose jobs into smaller tasks which are executed across a number of machines in order to achieve improved performance through parallelization. While such frameworks have seen substantial success in recent years, they also have their own set of challenges. Specifically, it has been established that achieving predictable execution within Cloud computing environments is problematic due to resource interference, scheduling and volatile network conditions [23]. This has resulted in significant challenges in provisioning real-time QoS within current Cloud computing systems. Furthermore, with the increased usage, system scale and application complexity, such behavior has been demonstrated to be increasingly important and more common place [3]. This is particularly true of Long Tail phenomena, which arise when these frameworks are deployed in larger-scale infrastructure. Long Tail phenomena can cause poor job execution due to abnormally slow parallel tasks known as stragglers. An example of such straggler tasks is shown in Figure 1 taken from a major production Cloud datacenter (for commercial reasons we cannot identify the name of the company). Such stragglers can impede a job's completion, as it is unable to complete until all its respective tasks are completed. Even after applying state-of-the-art straggler mitigation techniques, stragglers still execute on average eight times slower than the median task in a certain job, and increases the average job duration by 47% [14].

Stragglers can occur due to many reasons, including hardware heterogeneity [2], resource contention, background network traffic, I/O discord [3] and OS and application-level related sources [4][23]. A number of works focus on straggler data skew, categorized as either Map or Reduce skew, and further subdivided into partitioning skew, record size skew and computational skew [5][6][7]. How the distribution of input dataset can cause data skew - hence introducing stragglers into the system - is discussed further detailed in [8].

As the size of clusters and jobs continue to grow, the impact of stragglers increases dramatically. Such stragglers can substantially extend job execution time, thus impacting QoS and potentially a consumer's Service Level Agreement (SLA) [9]. Even rare performance abnormalities can affect a significant portion of all requests in large-scale distributed systems [2], and so addressing the Long Tail problem is critical in order to speed up job completion and improve system efficiency.

III. RELATED WORK

A. Long Tail Mitigation

Eliminating all sources of stragglers in large-scale systems is impractical, due to system scale and increasing use of multi-tenancy to collocate tasks within the same physical server As a result, a typical approach is to attempt to mitigate the impact of straggling tasks occurring within a system through the use of speculation. Initially proposed in by Dean et al. [1], this technique observes the progress of individual tasks and launches speculative copies (or backup copies) for tasks that

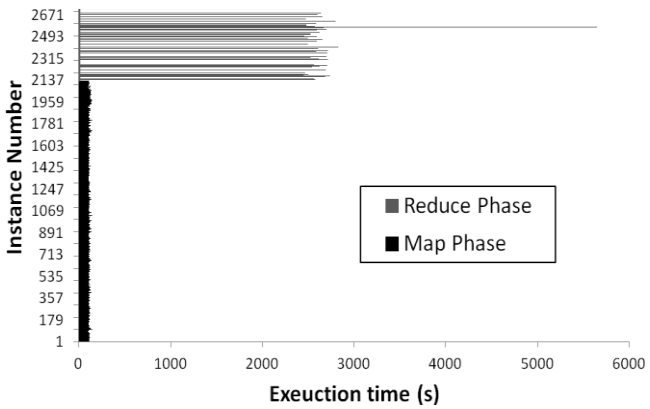

Fig 1. Example of straggler occurrence causing Long Tail within a job.

are running slower than average task execution at runtime. Such an approach assumes that the speculative copy should execute faster than the original straggling task, and is commonly deployed in many production clusters including Facebook, Google, Bing, and Yahoo. In recent years there have been a number of proposed speculative execution based methods: LATE [2] focuses on mitigation within heterogeneous environments, Mantri [9] leverages network congestion characteristics as well as preferentially replicating the output of tasks that are more likely to be lost or expensive to recompute, while GREST [11] leverages data locality of Map tasks when performing speculation. Dolly [14] is concerned exclusively with small jobs that contain less than 10 tasks; this characteristic enables the scheme to copy all jobs, and ignores its impact on resource consumption. GRASS [10] uses speculation between deadline bound and error bound. Furthermore, [12] introduces the concept of co-workers that assist with task re-execution, while MCP [13] accelerates the re-execution by choosing a suitable set of back up nodes.

B. Identification and Analytics

The identification of stragglers plays an important role in speculative execution, and its effectiveness is measured by identification accuracy and rapidness. However, current approaches will wait until a straggler has been identified from online processing [2][12][13] which typically occurs late in a task's execution lifecycle. An effective means to improve the rapidity of straggler identification is through the use of offline analytics and historical data to characterize the temporal patterns of Long Tail manifestations within a system for given tasks or nodes. There are several approaches that leverage historical data to support straggler speculation performance: SAMR [16] uses historical data to adjust temporal weightings for each execution stage when calculating task progress, while MCP [13] leverages historical data to select the most suitable backup nodes for speculative tasks to run. Furthermore, there are methods that use historical data to proactively avoid scenarios that cause stragglers: Wrangler [15] uses a statistical learning technique based on cluster resource utilization counters, while [17] uses historical data to identify nodes which cause stragglers. While several works have attempted to avoid the occurrence of straggler nodes [17][24], to our knowledge, no existing system has used historical data to specifically identify task stragglers.

From the related work, it is observable that there is need for an approach that can identify stragglers as quickly and accurately as possible. This is especially the case when considering jobs that are time sensitive or have temporal guarantees (through

SLAs, etc). Online and offline analytics are an effective means to achieve this; however, both face challenges when used in isolation. Firstly, the use of online analytics for identification can occur too late in task execution, still resulting in delayed job completion times, debilitating the system's ability to provision timely service to users. Second, offline analytics are predominantly applied for straggler avoidance within the system, which becomes less feasible when approaching systems at increasing scale (which can experience Long Tail phenomena through numerous scenarios). Therefore, there is a clear opportunity to integrate both online and offline analytic techniques together to improve our ability to identify Long Tail behavior in a manner that can preserve the temporal guarantees within a system.

IV. System Architecture

This paper proposes a method and tool for straggler identification to support the mitigation of Long Tail phenomena within large-scale distributed systems with much improved timeliness. A challenge in large-scale systems is to identify straggler manifestation, which can substantially impact job execution completion time. While our approach could be applied to a number of different types of distributed systems, in this work we focus on Cloud computing datacenters, a modern large-scale system that contain explicit (SLAs, QoS) and implicit (energy-efficiency, user experience) requirements for provisioning timely service to users.

In order to achieve our objective of improving the rapidity and accuracy of straggler identification, we propose an architecture (as shown in Figure 2) composed of two primary components: offline and online analytics. The offline analytics component data mines historical system traces in order to characterize and model task execution patterns and calculates a theoretical threshold that determines the boundary of non-straggler behavior for task execution at a given time interval. The online analytics monitors and compares task execution at runtime through the use of agents against models created from the offline analytics in order to identify task straggler behavior.

A. Offline Analytics

The offline analytics component of our approach is conceptualized within the Long Tail Analytics Engine, and is responsible for analyzing and modeling task behavior and straggler manifestation through the use of historical analysis. Specifically, this module is responsible for modeling task execution patterns as well as informing the online analysis for identifying straggler behavior at runtime. The module is composed of three sub-components:

Job profiler: This component is responsible for profiling and modeling different types of jobs and task execution patterns within the system. This is an important consideration as work in [19] demonstrates how tasks are capable of exhibiting different task execution lengths and resource consumption across a system. The method of profiling job execution patterns is dependent on the nature of tasks within the system, and can be performed using a number of techniques such as clusterization, and modeling task progress execution [18]. Figure 3 shows the latter approached applied to 500 Reduce tasks within a 50 node cluster. It is observable that the Reduce phase can be divided into multiple phases [1][2] which can be modeled through the use of linear and non-linear regression analysis. Such a technique makes it possible to calculate typical task execution progress over time for a given job

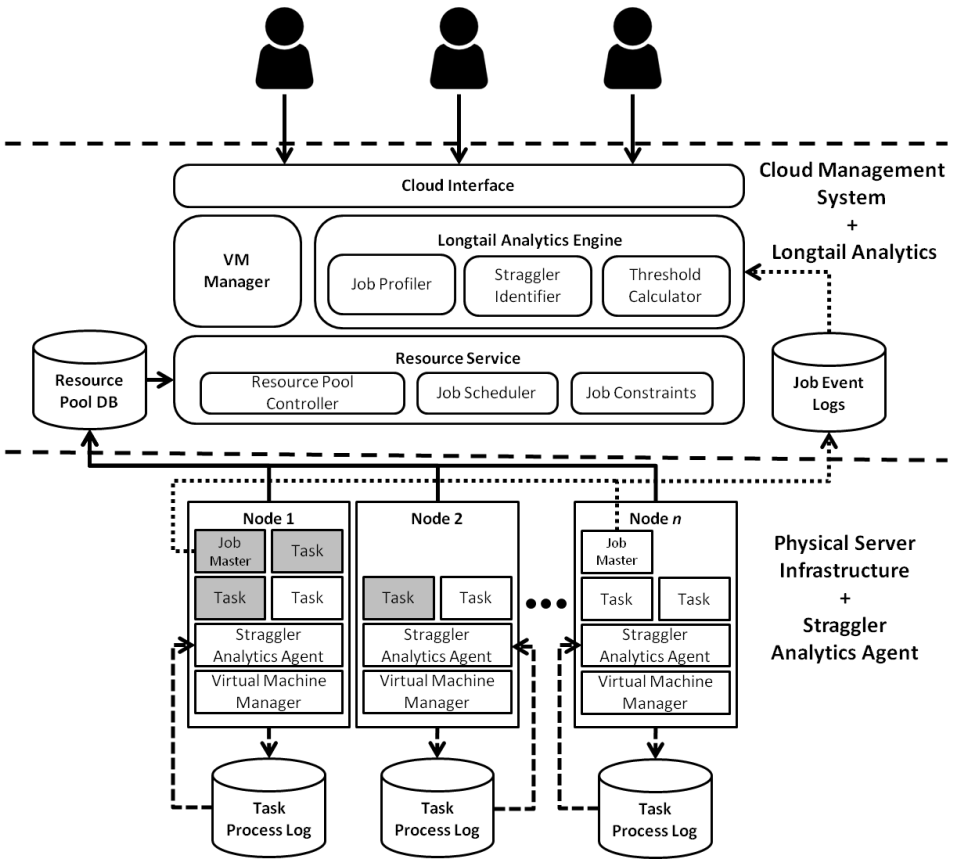

Fig 2. Cloud computing model with integrated Long Tail analytics engine and agent based analytics.

profile and provide statistical models which can be integrated into the online analytics agents.

Straggler Identifier: This component is responsible for quantifying the nature and impact of stragglers within the distributed system. Work in [3][7][9] have identified that there are numerous root cause for Long Tail as discussed in Section 2. Therefore, it is advantageous to analyze and identify the cause of stragglers which occur historically within a system in order to ascertain where developmental and mechanism effort should be applied for maximum effectiveness.

Threshold Calculator: The threshold calculator is responsible for generating models to identify potential task stragglers. This is achieved by exploiting the task execution patterns and models generated from the job profiler component to derive a (theoretical) minimum threshold for task progress at a certain time. Specifically, straggler threshold S is defined as minimum progress of task T_i completed at time t in relation to normal task progress Prog to avoid a task being flagged as a straggler. The difference between T_{iProg} and T_{iS} at time t is given by Diff, representing the maximum completion difference acceptable before straggler identification, and is expressed as a percentage configured by the provider.

To give a hypothetical example, if a model which expresses T_i over period t generated by the Job Profiler component is defined as shown in (1):

$$T_{iProg} = 0.01 + 0.15t \tag{1}$$

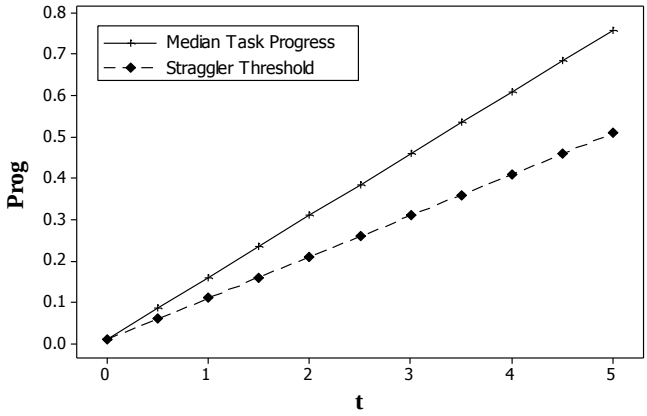

Fig 4. Representation of median task progress (T_{iProg}) and straggler threshold (T_{iS}) at time t.

and Diff is defined as task execution time 50% greater than median execution (a value commonly defined in the literature [7]), then the straggler threshold is expressed as shown in (2):

$$T_{iS} = 0.01 + 0.15 \cdot \frac{2}{3}t \tag{2}$$

As demonstrated in Figure 4, T_{iS} will equal T_{iProg} when t is 50% greater (thus, a task is identified as a straggler when the time taken to reach a specific progress score at time t is greater than 50% compared to typical task execution). The developed model generated from the offline component of the system are exploited by the online analytics at runtime to identify stragglers.

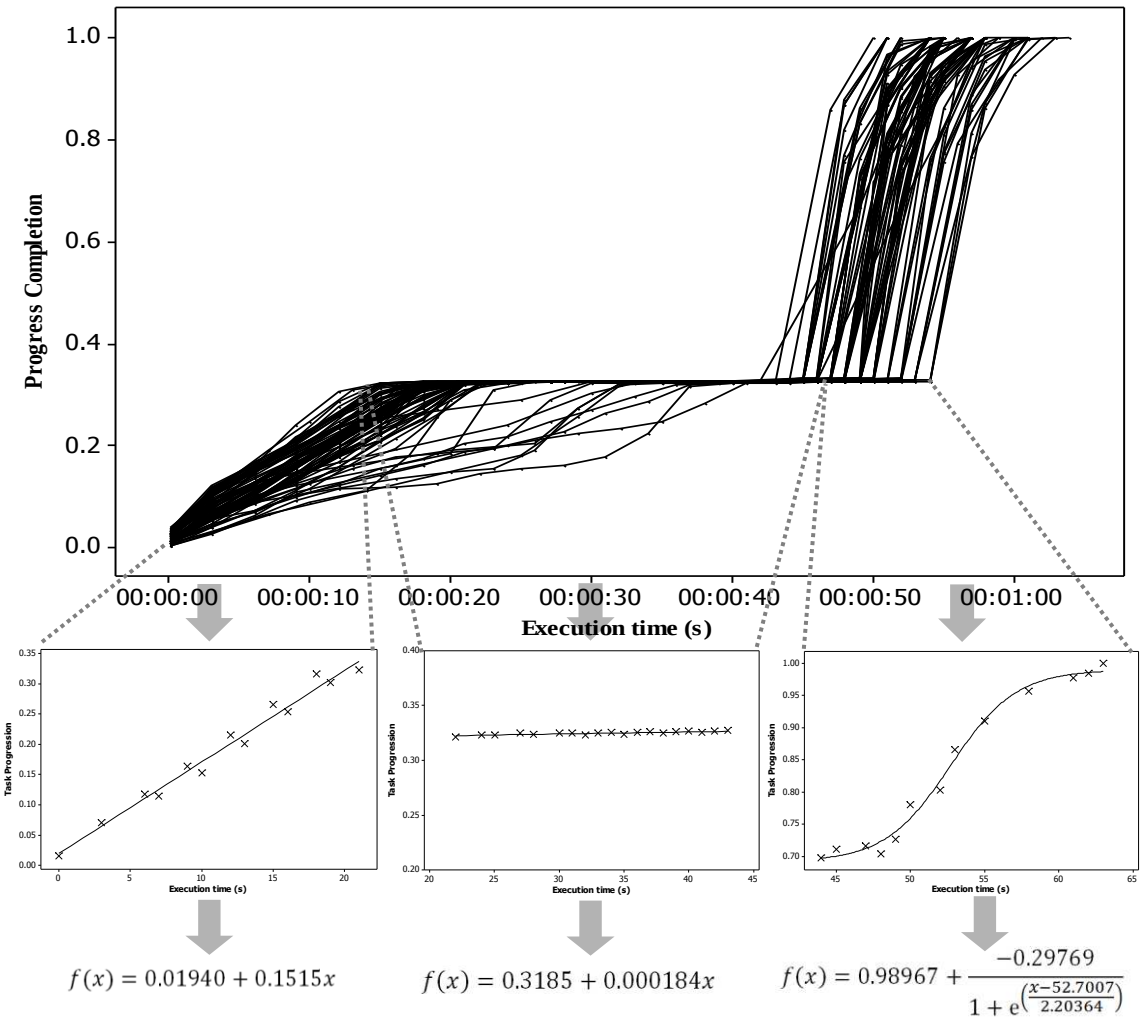

Fig 3. Example of converting empirical task progression into statistical models for 50 node cluster.

B. Online Analytics

The online analytics component is comprised of the **Straggler Analytics Agent** which resides on each individual physical server within the distributed system as shown in Figure 2. This is responsible for monitoring and analyzing task execution progress and identifying task stragglers at runtime. Upon task scheduling onto a server, each agent will periodically monitor and extract key parameters from progress logs generated by each task. Such parameters of interest includes the timestamp, time of task instantiation, current task progress score as well as additional parameters including data blocks transferred and download rate if applicable.

The online agent compares current task progress against the model produced by the offline analysis derived from the Long Tail Analytics Engine. Furthermore, the agents also compare the current progress of other tasks within the same job. The model derived from the offline analysis is of particular importance, as multiple stragglers within a job will result in increased Long Tail identification time when solely comparing task progress scores at runtime. If $T_{iProg} < T_{iS}$, as well as 50% smaller than the median task progression at t_i for its respective job, a task is identified as a straggler. Such an approach can encounter challenges in model sensitivity within the first time periods due to the short Euclidian distance between progress scores at the start of task execution. As a result, stragglers are identified if $T_{iProg} < T_{iS}$ consecutively n times, where n is defined by the provider.

V. Long Tail Analytics in Production Systems

This section fulfils two purposes: First, an empirical analysis of two large-scale Cloud datacenters is conducted in order to justify the impact of stragglers within production systems. Second, this section provides a practical example how the Straggler Identifier component operates within the system as discussed in Section 4. To facilitate this, we have analyzed two distinctive trace logs of large-scale production Cloud datecenters. The first trace log is Google's datacenter; a system composed of over 12,500 servers and millions of tasks over a period of 29 days, composed by a large number of different types of applications. The trace log is publicly available and can be found at [20]. The second trace log is Cloud datacenter B; large-scale production e-commerce system containing over 1,300,000 tasks and 2,800 servers.

In order to conduct a comprehensive analysis, jobs and their respective tasks are filtered to fulfil a specific criteria: Specifically, we are particularly interested in straggler manifestation within batch jobs, which is possible to derive given the characteristics of the job priority, job start and completion time in relation to task submission and completion, as well as resource characteristics (i.e. all tasks within a job have the same requested resources and are submitted fraction of timestamps apart from each other). Through this filtering criteria it is possible to identify 3043 jobs comprised of 252,950 tasks within Google and 875 jobs comprised of 1,223,879 tasks for Cloud datacenter B.

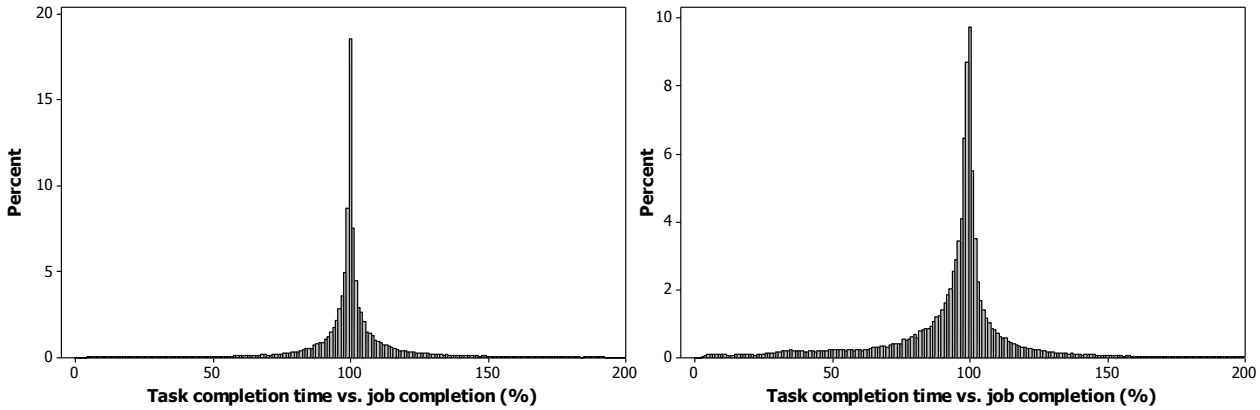

Fig 5. Google Datacenter task - job completion difference %
(a) median, (b) mean.

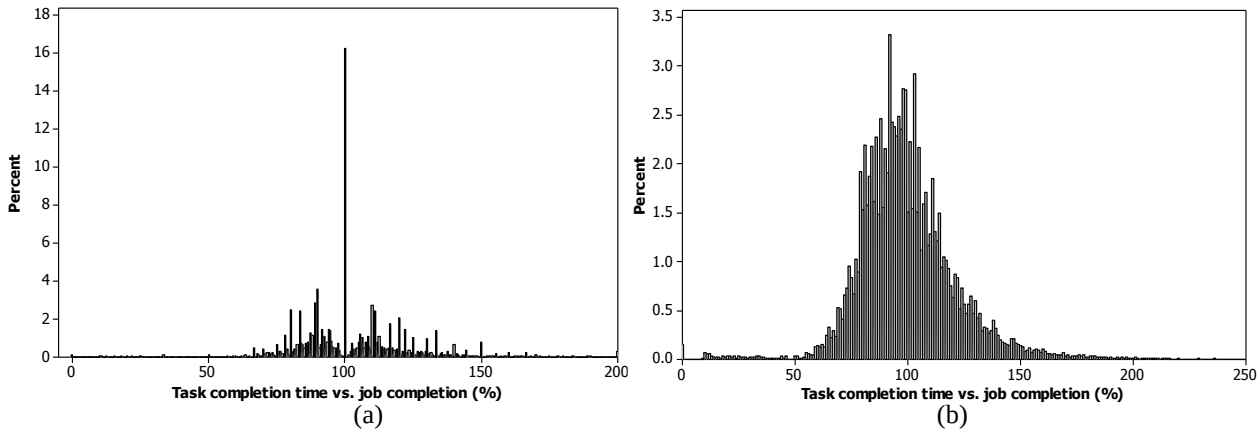

Fig 6. Cloud Datacenter B task - job completion difference %
(a) median, (b) mean.

TABLE 1. DATA SKEW STRAGGLERS IN PRODUCTION SYSTEMS.

	Google Datacenter		Cloud Datacenter B	
	Mean	**Median**	**Mean**	**Median**
Total tasks	252,950		1,233,879	
Task stragglers	11,210	16,543	33,322	42,925
Task stragglers %	4.43	6.54	2.70	3.48
Total jobs	3043		875	
Job stragglers	1081	1150	512	433
Job stragglers %	35.52	37.79	58.51	49.49

Figure 5 and 6 show the mean and median completion time of an individual task in comparison to respective tasks within the same job for Google datacenter and Cloud datacenter B, respectively. It is observable tasks exhibit similar percentages of completion time around 100% across both Cloud datacenters, with a small portion of tasks completing much earlier or later. Furthermore, we observe that studying the mean and median of task execution time is substantially different, most notable for Cloud datacenter B in Figure 6. This is predominantly caused by extremely fast and slow tasks significantly affecting the mean value for task completion. As a result, while existing literature use the mean execution time for defining task stragglers, we feel that the median task duration is a more sensible approach. This results in 6.54% and 3.48% of tasks to exhibit straggler behavior in Google datacenter and Cloud datacenter B, respectively when studying mean application execution.

While it might be intuitive to assume such a small portion of straggled tasks may cause limited impact on job performance, it actually results in significant impact to total job completion times, indicated by 37.79% and 49.49% of jobs being straggled within each datacenter, respectively. This is a result of jobs unable to complete until all its respective tasks (including stragglers) have completed execution.

Furthermore, the number of stragglers per server is also studied as shown in Figure 7(a) and 7(b) for Google datacenter and Cloud datacenter B, respectively. We observe that 19.9% and 99.78% of servers experience task stragglers across each system, and while at different proportions, exhibit a similarly weakly skewed distribution.

These observations indicate two points of interest. First, given the available trace log data from two production Cloud computing datacenters, stragglers caused by data skew appear to have significant impact in terms of timeliness of job execution within the system due to frequent occurrence of thousands of stragglers within the infrastructure, manifesting Long Tail behavior in 35-59% of total jobs. Second, it appears that node stragglers appear to have minimal impact due to light skew of straggler occurrence per node within the system. Such results highlights the need to study, identify and mitigate straggler behavior caused by data skew within tasks.

VI. EVALUATION

A. Experiment Setup

Experiments were conducted to evaluate the effectiveness of our method in terms of straggler identification accuracy and rapidity within real systems. To facilitate this, our method was implemented and evaluated by using a 50 node cluster comprised of 40 x quad-core Intel machines @ 3.40GHz CPU running CentOS. This system is used concurrently by other users for research and university services. The application use case for the system was Hive [22], a database management system which interfaces and translates user specified queries

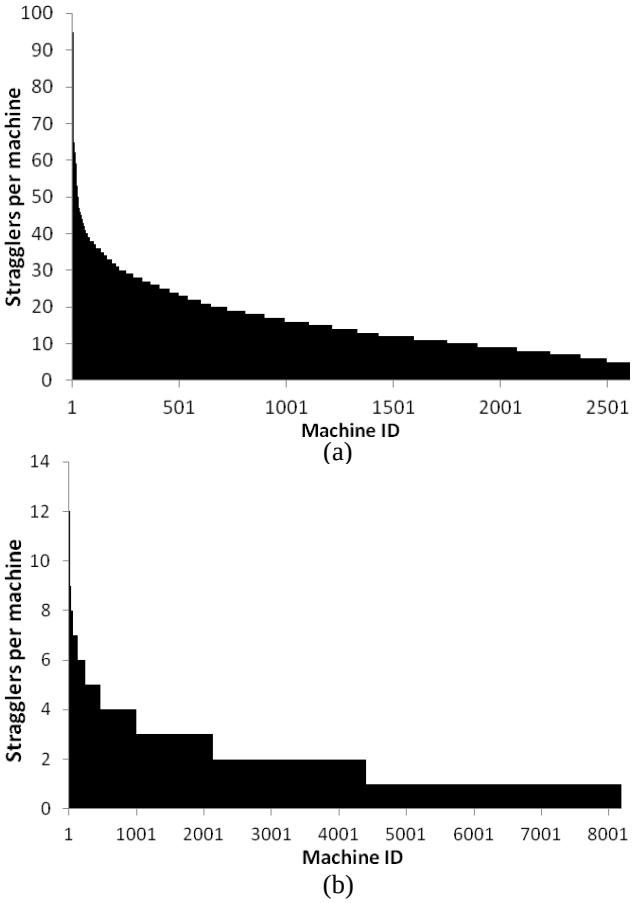

Fig 7. Comparison of filtered stragglers from (a) Google datacenter, (b) Anonymous e-commerce Cloud system.

into MapReduce tasks. Approximately 40 jobs comprised between 500-1000 Map tasks and 10-30 Reduce tasks were submitted into the cluster; each job was configured to vary in terms of data computation, (multiplication, CEIL and FLOOR functions), number of JOIN clauses between data tables, and the type of data attributes processed. Due to the severity of data skew in Cloud datacenters identified in Section 5, upon each submission there is a probability that the query will invoke stragglers caused by data skew in a number of tasks within the Reduce phase. This probability was configured to be 5%, reflecting similar values discovered in the empirical analysis discussed in Section 5, and was dictated by invoking queries which are known to cause data skew within Hive. Task straggler behavior was defined as task completion time 50% greater than the median task execution within a job in accordance to similar values reported and defined in [7]. From conducting initial experiments into mapping the proposed scientific model into a real-world implementation, the online analytic agents were configured to flag tasks exhibiting straggler behavior when $T_{iProg} < T_{iS}$ for 3 consecutive monitoring periods, and analyze progress scored 5 seconds after task execution has begun. Finally, online analytics agents were configured to monitor and compare current task progress against the models generated from Long Tail Analytics Engine at a time interval of one second.

B. Results

Figure 8 depicts the progress of Reduce execution over time, as well as highlights the generated threshold model and straggler identification for a given job. It is observable that after 30 seconds (approximately 50% task progress), there is a significant difference between normal task and straggled task

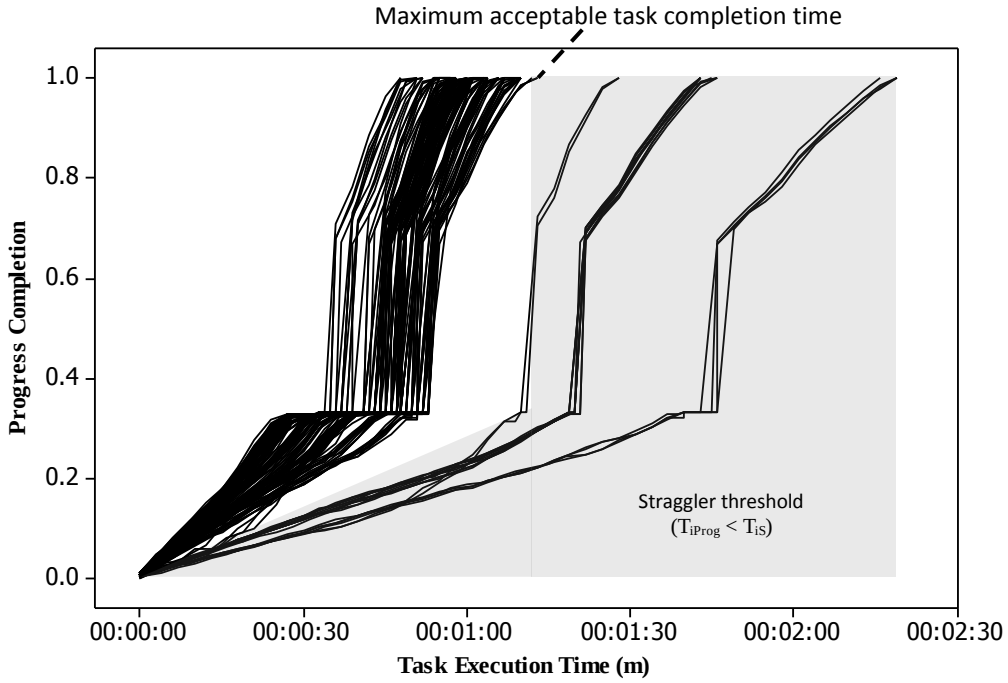

Fig 8. Task Progress with Long Tail Identification Analytics Engine.

progress patterns due to the input size pushed to an individual Reduce task becoming more distinct further into task execution. Through using statistical models generated using historical data from the offline analysis in conjunction with the online analytic agents, it is possible to identify over 98% of stragglers caused by data skew that are flagged on average 10.91% into a task's progress at runtime as shown in Table 2.

Furthermore, we observe a false positive rate of 1.59% which is caused by task progress of straggler and non-straggler tasks exhibiting similar progress scores at the start of execution as shown in Figure 8. This is predominantly caused by threshold sensitivity or interference from other users executing jobs on the same cluster. This result demonstrates the need for refinement of any future identification techniques that are configured to handle potentially different sensitivity levels for straggler identification at different time frames into a task's execution. While this result indicates high accuracy of straggler identification, from our experiments we observe that there is further refinement required for identifying true positive task stragglers less than 10% into a task's execution. Such refinement could be achieved through tuning Diff as well as data mining additional event parameters of interest from system logs (i.e. task process resource consumption, network usage, node location, etc.).

Moreover, we observe that a minority of tasks are flagged as stragglers early within their execution however finish relatively close to the boundary of acceptable task completion.

TABLE 2. STATISTICAL PROPERTIES OF DATA SKEW EXPERIMENTS.

Total Reduce tasks submitted	410
Total stragglers submitted (%)	6.45
True positive rate (%)	98.71
False positive rate (%)	1.59
Straggler progress detection (%)	10.91
CPU usage of agent per node	0.20%

This behavior highlights the potential issues of defining a fixed value for Long Tail phenomena (i.e. 50% greater than median task execution completion time), as tasks which complete just under the threshold will be flagged as false positives, however still substantially impede timely job completion. Such a result indicates that there is a need for a more intelligent metric for defining Long Tail phenomena. One such solution would entail transitioning away from a fixed temporal boundary as defined in [7] to a progressive boundary relative to the distance between task progress and median task completion time for straggler identification.

Finally, the online analytics agent are demonstrated to be lightweight, indicated by CPU usage of 0.2% usage for a fraction of a time frame periodically for each server, and no indication that it has significant effect on task progress execution depicted in Figure 8.

VII. CONCLUSION

In this paper, we have developed a method and tool for Long Tail identification in distributed systems that for the first time combines both offline analytics and online agent based monitoring/analytics in order to improve the timeliness of straggler identification. This work demonstrates that our approach is capable of identifying task stragglers caused by data skew rapidly and accurately at runtime, and can be integrated into state-of-the-art straggler mitigation techniques in order to enhance the temporal properties and timeliness of job execution. Furthermore, we have presented an empirical analysis of two large-scale production Cloud datacenters exemplifying the impact of stragglers caused by data skew. Such results provides key empirical insight into how job timeliness and performance is significantly degraded due to straggler occurrence. Our conclusions are listed as follows:

– Holistic usage of offline and online analytics is an effective means to identify Long Tail behavior at runtime. Through a novel approach of offline and online agent based analytics, we demonstrate through empirical experiments

that it is possible to identify 98% of task stragglers approximately 11% into a task's execution. Such results signify that our approach is capable of identifying task stragglers relatively early, and can substantially reduce the time to detect straggler behavior, hence improving the temporal properties of executing jobs when combined with state-of-the-art straggler mitigation techniques. Furthermore, our approach was demonstrated to scale to 50 physical machines whilst utilizing minimal resources due to the agent based architecture, and is likely to operate sufficiently within larger-scale systems.

- Data skew that occurs in a small subset of tasks significantly impacts job completion time. Our empirical analysis of two production distributed systems composed of thousands of nodes demonstrates that 4% and 6% of total task stragglers cause Long Tail behavior to manifest within 37 – 49% of total jobs, exemplifying the challenges large-scale systems face. With the evolving trend of computing systems growing in complexity and scale, such findings demonstrate the significant threat that Long Tail phenomena imposes towards guaranteeing application timeliness and performance within next generation systems. This work highlights and discusses the urgent need for research that addresses this challenge and attempts to limit its impact on efficient system operation.
- Challenges in straggler identification due to data skew occur primarily at beginning of task execution. We discover that there are potential challenges in detecting straggler behavior at runtime within the very first time periods into job execution. This is reflected in experiments by a straggler identification false positive of approximately 2%. This is a result of task progress scores being extremely similar at the beginning of task execution, signifying sensitivity of identification could potentially change at different periods through a task's execution. As a result, while our approach is effective for identifying stragglers 11% into task execution, refinement is required in order to achieve straggler identification within the first few seconds of task execution.

Future work includes integration of our approach into established Long Tail mitigation techniques including speculative speculation and similar techniques to discover whether we can achieve substantial gains in job completion timeliness and system QoS. Furthermore, while data skew has been demonstrated to significantly affect temporal behavior of jobs, there exist other causes of Long Tail phenomena including performance interference and garbage collection. Therefore, there is an opportunity to extend our approach allowing it to identify different causes of stragglers at runtime. Finally, current definition of stragglers are indicated by completion times greater than 50% of the mean task execution; such a definition restricts the usefulness for subsequent Long Tail identification and mitigation for tasks which complete prior to this (i.e. 145 - 149%). As a result, we plan to develop a more effective straggler identification model which factors in task completion times within the context of job QoS specified by a user or provider.

Acknowledgments

The work is supported in part by the National Basic Research Program of China (973) (No.2011CB302602), the U.K. EPSRC WRG platform project (No. EP/F057644/1), and other EPSRC and RC grants.

References

[1] J.Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communications of the ACM 51.1, pp. 107-113, 2008.

[2] M.Zaharia, A. Konwinski, A.D. Joseph, R. Katz, I. Stoica, "Improving MapReduce Performance in Heterogeneous Environments." in Proc. of the 8th USENIX conference on Operating systems design and implementation (ODSI), pp 29-42, 2008.

[3] J. Dean, L. A. Barroso, "The Tail at Scale." Communications of the ACM 56.2, pp.74-80, 2013.

[4] J. Li, N. K. Sharma, D. R. K. Ports, S. D. Gribble, "Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency." ACM Symposium on Cloud Computing (SOCC), 2014.

[5] Y. Kwon, M. Balazinska, B. Howe, J. Rolia, "A Study of Skew in MapReduce Applications." Open Cirrus Summit, 2011.

[6] Y. Kwon, M. Balazinska, B. Howe, J. Rolia, "Skewtune: Mitigating Skew in Mapreduce Applications." in Proc. of ACM SIGMOD International Conference on Management of Data, pp. 25-36, 2012.

[7] J. Rosen, B. Zhao. "Fine-Grained Micro-Tasks for MapReduce Skew-Handling.", White Paper, University of Berkeley, 2012.

[8] J. Lin, "The Curse of zipf and Limits to Parallelization: A look at the Stragglers Problem in Mapreduce." In Proc. of 7th Workshop on Large-Scale Distributed Systems for Information Retrieval, Vol. 1, 2009.

[9] G. Ananthanarayanan, et al. "Refining in the Outliers in Map-Reduce Clusters using Mantri." in Proc. of USENIX conference on Operating systems design and implementation (ODSI) Vol 10, 2010.

[10] G. Ananthanarayanan, et al. "GRASS: Trimming Stragglers in Approximation Analytics." in Proc. of 11th USENIX Conference on Networked Systems Design and Implementation, pp. 289-302, 2014.

[11] L. Lei, T. Wo, C. Hu, "CREST: Towards Fast Speculation of Straggler Tasks in MapReduce." in Proc. of IEEE 8th International Conference on e-Business Engineering (ICEBE), pp. 311-316, 2011.

[12] S.W. Huang, T.C. Huang, S.R. Lyu, C.K. Shieh, Y.S. Chou, "Improving Speculative Execution Performance with Coworker for Cloud Computing." In Proc. of 17th IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 1004-1009, 2011.

[13] Q. Chen, C. Qi, X. Zhen "Improving MapReduce Performance using Smart Speculative Execution Strategy.", in Proc. of IEEE Transactions on Computers, Issue No. 4, pp. 954-967,2013.

[14] G. Ananthanarayanan, A. Ghodsi, S. Shenker, I.Stoica, "Effective Straggler Mitigation: Attack of the Clones." In Proc. of the 10th USENIX conference on Networked Systems Design and Implementation (NSDI). Vol. 13, pp. 185-198, 2013.

[15] N.J. Yadwadkar, G. Ananthanarayanan, R. Katz, "Wrangler: Predictable and Faster Jobs using Fewer Resources." in Proc. of the ACM Symposium on Cloud Computing, pp. 1-14, 2014.

[16] Q. Chen, D. Zhang, M. Guo, Q. Deng, "SAMR: A Self-adaptive MapReduce Scheduling Algorithm in Heterogeneous Environment." in Proc. of IEEE 10th International Conferience on Computer and Information Technology (CIT), pp. 2736-2743, 2010.

[17] N.J. Yadwadkar, W. Choi. "Proactive Straggler Avoidance using Machine Learning.", White paper, University of Berkeley, 2012.

[18] I.S. Moreno, P. Garraghan, P. Townend, J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models" in Proc. of IEEE International Symposium of Service-Oriented System Engineering (SOSE), pp. 49-60, 2013.

[19] I. Solis Moreno, P. Garraghan, P. Townend, J. Xu "Analysis, Modeling and Simulation of Workload Patterns in a Large-Scale Utility Cloud", IEEE Transactions on Cloud Computing, vol.2, no.2, pp.208-221, 2014.

[20] Google. Google Cluster Data V2. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1

[21] C. Reiss, J. Wilkes, "Google Cluster-Usage Traces: Format and Schema," Google Inc., White Paper, 2011.

[22] A. Thusoo, et al. "Hive: a Warehousing Solution over a MapReduce Framework.", Processing of VLDB Endowment, Vol. 2, Issue 2, pp 1626-1629, 2009.

[23] M. García-Valls, T. Cucinotta, C. Lu, "Challenges in Real-time Virtualization and Predictable Cloud Computing." Journal of Systems Architecture 60, pp.726-740, 2014.

[24] Y.Xu, Z. Musgrave, B. Noble, M. Bailey. "Bobtail: Avoiding Long Tails in the Cloud." in Proc. of 10th USENIX conference on Networked Systems Design and Implementation, pp. 329-342. 2015.