On Understanding the Energy Impact of Speculative Execution in Hadoop
Tien-Dat Phan, Shadi Ibrahim, Gabriel Antoniu, Luc Bougé

To cite this version:
Tien-Dat Phan, Shadi Ibrahim, Gabriel Antoniu, Luc Bougé. On Understanding the Energy Impact of Speculative Execution in Hadoop. GreenCom’15-The 2015 IEEE International Conference on Green Computing and Communications, Dec 2015, Sydney, Australia. hal-01238055

HAL Id: hal-01238055
https://inria.hal.science/hal-01238055
Submitted on 4 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.
L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
On Understanding the Energy Impact of Speculative Execution in Hadoop

Tien-Dat Phan  
ENS Rennes / IRISA  
Rennes, France  
tien-dat.phan@irisa.fr

Shadi Ibrahim  
Inria Rennes - Bretagne Atlantique  
Rennes, France  
shadi.ibrahim@inria.fr

Gabriel Antoniu  
Inria Rennes - Bretagne Atlantique  
Rennes, France  
gabriel.antoniu@inria.fr

Luc Bougé  
ENS Rennes / IRISA  
Rennes, France  
luc.bouge@ens-rennes.fr

Abstract—Hadoop emerged as an important system for large-scale data analysis. Speculative execution is a key feature in Hadoop that is extensively leveraged in clouds: it is used to mask slow tasks (i.e., stragglers) — resulted from resource contention and heterogeneity in clouds — by launching speculative task copies on other machines. However, speculative execution is not cost-free and may result in performance degradation and extra resource and energy consumption. While prior literature has been dedicated to improving stragglers detection to cope with the inevitable heterogeneity in clouds, little work is focusing on understanding the implications of speculative execution on the performance and energy consumption in Hadoop cluster. In this paper, we have designed a set of experiments to evaluate the impact of speculative execution on the performance and energy consumption of Hadoop in homo- and heterogeneous environments. Our studies reveal that speculative execution may sometimes reduce, sometimes increase the energy consumption of Hadoop clusters. This strongly depends on the reduction in the execution time of MapReduce applications and on the extra power consumption introduced by speculative execution. Moreover, we show that the extra power consumption varies in-between applications and is contributed to by three main factors: the duration of speculative tasks, the idle time, and the allocation of speculative tasks. To the best of our knowledge, our work provides the first deep look into the energy efficiency of speculative execution in Hadoop.

Keywords—Cloud; MapReduce; Hadoop; stragglers, speculation; energy;

I. INTRODUCTION

MapReduce has become a prominent framework for large-scale data analysis on clouds. The popular open source implementation of MapReduce, Hadoop [1], was developed primarily by Yahoo!, where it processes hundreds of terabytes of data on at least 10,000 cores, and is now used by other companies, including Facebook, Amazon, Last.fm, and the New York Times [2].

Hadoop was designed with software/hardware failure in mind. In particular, Hadoop tolerates machine failures (crash failures) by re-executing all the tasks of the failed machine by the virtue of data replication. Furthermore, in order to mask the effect of stragglers (tasks performing relatively slower than other tasks, due to the variation in CPU availability, network traffic or I/O contention), Hadoop re-launches other copies of such tasks on other machines. Since Hadoop job’s response time is dominated by the slowest tasks instance times, stragglers can severely impede the job execution, resulting in longer response time of the whole job. It is therefore essential for Hadoop to identify these stragglers and run speculative copies of them and, thus, lower the response times.

Though benefits exist (e.g., Google has noted that speculative execution can improve job response times in Google cluster by 44% [3]), launching speculative tasks can be an expensive operation due to the extra resource consumption they imply. Speculative execution will obviously result in higher energy consumption which is a critical concern in Hadoop clusters. Especially, given the ever growing size of data (i.e., Big Data) and increasingly high energy cost of operating datacenters which process this data (e.g., the annual electricity usage and bill of Google are over 1,120 GWh and $67 M, and they are over 600 GWh and $36 M for Microsoft [4]).

The increasing trend towards Big Data processing in the clouds [5, 6], the inevitable heterogeneity and dynamicity of cloud resources [7, 8], and the proliferation of different MapReduce applications [9–11] elevate straggler’s mitigation to a key issue in Hadoop. Hadoop clusters will encounter more stragglers and more speculative tasks will be launched, hence, higher energy consumption. Even worse, speculative tasks may not succeed, due to wrong or late detection. As shown in Figure 1, roughly 50% of the jobs launch the speculative copies which constitute on average 10% of the total tasks of submitted jobs. Among which less than 7% were successful.

Many recent work has been proposed to reduce the energy consumption in Hadoop cluster through the Dynamic Voltage Frequency Scaling (DVFS) support in hardware [12, 13], data-layout approaches [14, 15], resource consolidation [16, 17], virtualization [18], or by exploiting green energy [19]. But, to the best of our knowledge, there is no work on understanding
the implications of speculative execution on the performance and energy consumption in Hadoop cluster.

The prevalence of speculative execution and the growing cost of power bills and the high carbon emission make it imperative to understand this impact. To this end, a series of experiments have been conducted to explore the implications of different factors including resource heterogeneity of Hadoop cluster and the characteristics of MapReduce applications on the energy consumption. We do so through deploying Hadoop on the Grid’5000 platform [20], powered by two power distribution units (PDUs). Each node is mapped to an outlet, thus we can export fine-grained power monitoring. Our results reveal that while speculative execution increases the energy consumption in a homogeneous environment due to the launching of unnecessary speculative tasks, it may reduce the energy consumption of Hadoop in a heterogeneous environment. This reduction in energy depends on the reduction in the execution time of MapReduce applications and on the extra power consumption introduced by speculative execution. The primary contributions of this paper are as follows:

• **Speculation: to enable or to disable?** Similar to what Google has reported in [3], we find that speculative execution is an important feature in Hadoop clusters and it can significantly improve the running time of MapReduce applications when stragglers occur. Moreover, we show that the current speculative execution in Hadoop is not accurate and may lead to excessive unnecessary speculative execution and therefore results in longer execution time due to the extra resource contention.

• **A detailed study of the performance and power cost of speculative execution in Hadoop.** We find that the energy cost of speculative execution is proportional to the increase in the power consumption — caused by speculative tasks — in the cluster. This extra power consumption varies in-between applications and is contributed to by three main factors including the duration of speculative tasks (i.e., extra resource occupation), the idle time, and the allocation of speculative tasks. While extra slot occupation and idle time are the major contributors to extra power cost of speculative execution when running I/O-bound applications, they have less impact on the extra power cost of speculative execution when running CPU-intensive applications.

• **A microscopic analysis to illustrate the trade-off between performance and energy consumption when scheduling speculative tasks.** We find a clear trade-off between performance and power consumption when scheduling a speculative task. While launching speculative copies on idle nodes or nodes with a small number of running tasks results in lower average task runtime but leads to higher power consumption per node, launching speculative copies on nodes with a larger number of running tasks results in higher average task runtime but lower power consumption per node.

It is important to note that the work we present here is not limited to Hadoop and can be applied to different MapReduce implementations that are featured with speculative execution (e.g., Spark [21]).

**Paper Organization.** The rest of this paper is organized as follows: Section 2 briefly presents Hadoop and speculative execution in Hadoop, and discusses the related work. Section 3 describes the overview of our methodologies, followed by the experimental results in Section 4, 5 and 6. Finally, we conclude the paper and propose our future work in section 7.

## II. BACKGROUND AND RELATED WORK

In this section, we briefly introduce Hadoop, zoom in on the speculative mechanism in Hadoop and then discuss the prior research work on mitigating stragglers and energy management in Hadoop clusters.

### A. Hadoop Architecture

The Hadoop MapReduce is a software framework for distributed processing of large data sets on compute clusters [1, 22]. It runs on the top of the Hadoop distributed file system (HDFS). The system uses a master/slave architecture. The master, called JobTracker (JT) performs several operations: (a) it queries the NameNode (i.e., the master node of HDFS) for the block locations; (b) considering the information retrieved by the NameNode, it schedules the tasks on the slaves, called TaskTrackers (TT); (c) it monitors the success and failures of the tasks. Hadoop adopts a resource management model based on slots to represent the capacity of a cluster: each worker in a Hadoop cluster is configured to use a fixed number of map slots and reduce slots in which it can run tasks.

#### B. Speculative execution in Hadoop

In Hadoop, each TaskTracker sends a heartbeat to the JobTracker (every 3 seconds by default) reporting free slots (for running map or reduce tasks), and the progress scores of its running tasks (the fraction of each task’s total work that has been done on a 0-to-1 scale). The JobTracker will process the heartbeats and will respond by assigning a new task to the TaskTracker [23]. This will keep a track on the progress scores of all the running tasks and allow the job tracker to detect the stragglers by filtering the tasks which have a progress score smaller than the average - 0.2 (default Hadoop). A task \( t_s \) is labeled as a straggler if and only if:

\[
PS_s < \left( \frac{\sum_{i=2}^{k} PS_i}{2} - 0.2 \right)
\]

where \( PS_s \) is the progress score of the examined task and \( PS_i \) are the progress scores of all completed and ongoing tasks within the same category (map tasks or reduce tasks). Once stragglers are detected, Hadoop sorts these tasks accordingly to their starting times. Thus, the earliest-launched straggler will be considered first whenever there are available slots for launching a speculative copy.

### C. Related Work

While there have been many research efforts on mitigating stragglers by the means of speculative execution and energy consumption in Hadoop systems, but none of them has addressed speculative execution and energy as a whole.

#### Mitigating stragglers in Hadoop

Straggler’s mitigation in Hadoop has attracted a lot of attention in the last few years. Most work has focused on investigating and improving stragglers detection in Hadoop in the cloud [8, 24–26]. Zaharia et al. [8] have reported that network heterogeneity can cause
80% increase of the needless reduce speculation. Accordingly, an optimized solution based on progress rate is proposed, named Last Approximate Task to Execute (LATE), which alleviates the launching of needless speculation caused by network heterogeneity. Ananthanarayanan et al. [25] introduce Mantri, a resource-aware straggler handing mechanism. Mantri schedules speculative tasks only when there is a fair chance to reduce the tasks’ execution time with a low resource consumption. Moreover, Mantri handles the stragglers at an early stage of the execution by killing ongoing tasks to free up resources to launch speculative tasks if the chance of saving the execution time is high. Considering that the majority of jobs in production Hadoop clusters are small jobs, Ananthanarayanan et al. [27] present Dolly, a new approach to handle stragglers by launching multiple copies (i.e., clone) of tasks belonging to small jobs. After the first clone of a task is completed, the other clones will be killed in order to free the resources. By cloning the whole job, Dolly produces a small, extra resource consumption, but results in a significant performance improvement. In contrast, we focus on energy consumption: we aim to provide an in-depth study of the impact of speculative execution on the energy footprint of Hadoop clusters.

Energy management in Hadoop clusters. Energy consumption in the Hadoop clusters is an issue of extremely high importance. There have been several studies on evaluating and improving the MapReduce energy consumption in data-centers and clouds. Many of these studies focus on power-aware data-layout techniques [14–16, 28–30], which allow servers to be turned off without affecting data availability. GreenHadoop [14] separates the HDFS cluster into hot and cold zones and places the new or high-access data in the hot zone. Servers in the cold zone are transitioned to the power-saving mode and data are not replicated, thus only the server hosting the data will be woken up upon future access. Chen et al. [31] analyze how MapReduce parameters affect energy efficiency and discuss the computation versus I/O tradeoffs when using data compression in MapReduce clusters in terms of energy efficiency [32]. Chen et al. [33] present the Berkeley Energy Efficient MapReduce (BEEMR), an energy efficient MapReduce workload manager motivated by empirical analysis of real-life MapReduce with Interactive Analysis (MIA) traces at Facebook. They show that interactive jobs operate on just a small fraction of the data, and thus can be served by a small pool of dedicated machines with full power, while the less time-sensitive jobs can run in a batch fashion on the rest of the cluster. Recently, Ibrahim et al. [34] present a systematic analysis on the impact of different DVFS governors on the energy consumption of Hadoop cluster. Goiri et al. [19] present GreenHadoop, a MapReduce framework for a data-center powered by renewable green sources of energy (e.g. solar or wind) and the electrical grid (as a backup). GreenHadoop schedules MapReduce jobs when green energy is available and only uses brown energy to avoid time violations. In contrast to related work, this is the first study that analyzes and shows how speculative execution can affect the energy consumption of Hadoop cluster.

III. METHODOLOGY OVERVIEW

The experimental investigation conducted in this paper focuses on exploring the implications of speculative execution on the energy consumption of Hadoop cluster under different workloads. Hereafter, we describe the experimental environment: the platform, the used benchmarks, and deployment setup.

A. Platform

The experiments were carried out on the Grid’5000 [20] testbed. The Grid’5000 project provides the research community with a highly-configurable infrastructure that enables users to perform experiments at large scales. The platform is spread over 10 geographical sites. For our experiments, we used nodes belonging to the Nancy site of the Grid’5000. These nodes are outfitted with a 4-core Intel 2.53 GHz CPU and 16 GB of RAM. Intra-cluster communication is done through a 1 Gbps Ethernet network. Only 40 nodes of the Nancy site are equipped with power monitoring hardware consisting of 2 Power Distribution Units (PDUs), each hosting 20 outlets. Since each node is mapped to a specific outlet, we are able to acquire coarse and fine-grained power monitoring information using the Simple Network Management Protocol (SNMP). It is important to state that Grid’5000 allows us to create an isolated environment in order to have full control over the experiments and the obtained results.

B. Benchmarks

MapReduce applications are typically categorized as CPU-intensive, I/O bound, or both. For our analysis, we selected two applications that are commonly used for benchmarking MapReduce frameworks: distributed WordCount and distributed Sort. Of these 2 benchmarks, distributed WordCount is CPU-intensive, while distributed Sort is I/O bound since it generates significantly more output data.

Real-life application. In addition to the two aforementioned benchmarks, we select CloudBurst. CloudBurst is a MapReduce application designed to facilitate biological analysis. It leverages an optimized algorithm for mapping next-generation sequence data to the human genome and other reference genomes. CloudBurst spends most of the time in the reduce phase to analyze the different mapping possibilities between the input and the reference genomes data. During reduce phases, this application mainly consumes CPU resources. In our experiments, we use the human genome sequence produced by Genome Reference Consortium\(^1\) as the input data for CloudBurst application (human genome dataset is available here\(^2\)). Table I summarizes the characteristics and the configurations of the three aforementioned applications.

<table>
<thead>
<tr>
<th>Application</th>
<th>CloudBurst</th>
<th>Sort</th>
<th>WordCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominating Phase</td>
<td>reduce</td>
<td>shuffle</td>
<td>map</td>
</tr>
<tr>
<td>Resources</td>
<td>CPU</td>
<td>Network</td>
<td>CPU</td>
</tr>
<tr>
<td>Input size</td>
<td>100 MB</td>
<td>24.5 GB</td>
<td>24.6 GB</td>
</tr>
<tr>
<td>Output size</td>
<td>9.7 GB</td>
<td>24.5 GB</td>
<td>200 MB</td>
</tr>
</tbody>
</table>

Table I presents the various sizes of input and output data for the three applications used.

C. Hadoop deployment

On the testbed described in Section III-A, we configured and deployed a Hadoop cluster using the Hadoop 1.2.1 stable

\(^1\)http://www.ncbi.nlm.nih.gov/projects/assembly/grc/

\(^2\)http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/
version [1]. The Hadoop instance consists of 21 nodes to serve as both datanodes and tasktrackers, among which, one node was also configured to serve as namenode and jobtracker. The tasktrackers were configured with 8 slots for running map tasks and 2 slots for executing reduce tasks. At the level of HDFS, we used the default chunk size of 64 MB and the default replication factor of 3 for the input and output data. For CloudBurst application, since it is a reduce-intensive application and requires mainly the CPU resources, each tasktracker was configured with 8 reduce slots.

Heterogeneous Environment. Stragglers are mainly caused by the resource contention and heterogeneity in Hadoop clusters (e.g., VM placements and application co-locations). In order to produce repeatable heterogeneous environments, we created two heterogeneous Hadoop clusters composed of the machines described in Section III-A. In the first cluster, we vary the number of active cores per node to 1-core, 2-core, 3-core and 4-core. We divided the cluster into four groups. Each group consists of 25% of the total cluster nodes and configured to use 1 core, 2 cores, 3 cores, and 4 cores. In the second cluster, we vary the utilized network bandwidth to 25%, 50%, 75% and 100% of the available network bandwidth (1 Gbps in our testbed). Similarly, we divide the cluster into four groups and the nodes in each group are configured with four aforementioned network bandwidths. While tasks within the first cluster will exhibit variable access to the disk resources when reading and writing data, according to the number of the tasks per node (in our experiments, 2 map slots per core), tasks in the second cluster will exhibit different network access patterns according to the node assigned network bandwidth. We run CloudBurst and WordCount on the first cluster and Sort on the second one.

Note. The total time used to conduct our experiments exceeded 40 hours on 21 nodes in Grid’5000. Each experiment was repeated three times and the average values are used in the subsequent analysis.

IV. MACROSCOPIC ANALYSIS

In this section, we provide a high-level analysis of the experimental results we obtained. Our goal is to study the impact of speculative execution on the energy consumption in Hadoop cluster when running the three aforementioned applications in both homo- and heterogeneous environments.

Figure 2(b) depicts the total energy consumption of the three applications when enabling and disabling speculation in homogeneous environments. The total energy consumption of Hadoop cluster when enabling speculation increases by 9.8%,
for CloudBurst, 4.4% map speculative tasks for Sort, and 13.1% map speculative tasks for WordCount. But, the ratios of successful speculative copies are very low for the three applications. To understand these phenomena, we will discuss the results of each application separately.

CloudBurst application. CloudBurst is a reduce-intensive application and the execution time is dominated by the reduce phase. CloudBurst exhibits an explicit skew between different reduce tasks (a similar observation is reported in [35]). Hadoop blindly considers long-running reduce tasks as stragglers and therefore launches unnecessary speculative copies. As a result, Hadoop does not reduce the variation in reduce task runtimes (as shown in Figure 4(b)).

Sort application. Sort is a shuffle-intensive application and the network load strongly affects the execution time. As shown in Figure 4(b), the gap between the slowest reduce tasks and average task runtimes is relatively small. Consequently no reduce speculative copy is launched. On the other hand, the gap between the slowest map tasks and average task runtimes is big. This is mainly due to the non-local map tasks (i.e., non-local map tasks take longer time to complete because they need to fetch their input data from other nodes). Moreover, non-local map tasks cause a big variation in map task runtimes (similar observations are reported in [36]) according to the network load and the progress of the shuffle phase. Hadoop considers long running tasks which is a result of the non-local map tasks as stragglers and launches other copies of them. However, 75% of the launched speculative copies are unsuccessful. Even worse, the resulted network contention leads to longer shuffle (reduce) runtime (as shown in Figure 4(b)), and ends up in longer execution time of the whole application.

WordCount application. The execution time of WordCount is dominated by the map phase. The gap between the slowest map and reduce tasks and average task runtimes is relatively small (as shown in Figure 4(a) and Figure 4(b)). Consequently the number of speculative copies is relatively small. However, these speculative copies are launched as backup copies of non-local map tasks (Hadoop considers the non-local map tasks as stragglers).

In summary, we observe that:

- Speculation is triggered in Hadoop based on a too simplistic criterion, which does not consider the root cause of the variation in task runtimes. We find that reduce skew and non-local map tasks can lead to excessive unnecessary speculative execution.
- Unfortunately, the unnecessary speculative tasks may slow down other running tasks as they will compete for the resources and — in some cases — may result in a performance degradation of the whole application.

A common trend can be observed: speculative execution leads to higher energy consumption in homogeneous environment. Hereafter, we present a detailed comparative discussion of the various running applications.

CloudBurst vs. WordCount Both applications are cpu-intensive applications and their execution times are not impacted by speculation. As shown in Figure 6(a), the average power consumption of a node increases by 11% (from 84.14 to 93.59 Watt) and 12% (from 87.37 to 97.92 Watt) for CloudBurst and WordCount applications when enabling speculation, respectively. This is unexpected as the cluster resources which are occupied by speculative copies during 18% of the execution times of CloudBurst and only 8% of the execution times of WordCount (higher slot occupation will result in higher average power consumption), as shown in Figure 8. Furthermore, given that the idle time when running CloudBurst decreased by 25% while it decreases by 12% when running WordCount (as
Speculation

WordCount

Sort

(b) The average and longest reduce task runtimes

Fig. 7. The average and longest task runtime in heterogeneous environment

(a) Homogeneous environment

(b) Heterogeneous environment

Fig. 8. Extra slot occupation due to speculative copies in homogeneous environment

Fig. 9. The total idle time when speculation is enabled

Speculative execution results in a significant reduction in the energy consumption of Hadoop cluster when running CloudBurst. We observe 28.67% energy reduction (as shown in Figure 3(b)). The significant reduction in execution time is the major contributor to this energy reduction (i.e., the execution time is decreased by 47.6% as shown in Figure 3(a)). This is due to the high ratio of successful speculative copies (see Figure 5(b)); the ratio of successful speculative copies is 54.5% and 70.2% for map tasks and reduce tasks, respectively. More importantly, these successful speculative copies improve the average task runtimes of reduce tasks (the average task runtime is decreased by 32.7%) and reduce the runtime of the slowest task by 48.7% (see Figure 7(b)). However, we can still see the natural skew-reduce issue: the gap between the longest reduce task and the average reduce task runtime is almost the same as in homogeneous environment (see Section VI).

B. Zoom in on the energy impact of speculative execution

Speculative execution results in a significant reduction in the energy consumption of Hadoop cluster when running CloudBurst. We observe 28.67% energy reduction (as shown in Figure 3(b)). The significant reduction in execution time is the major contributor to this energy reduction (i.e., the execution time is decreased by 47.6% as shown in Figure 3(a)). This is due to the high ratio of successful speculative copies (see Figure 5(b)); the ratio of successful speculative copies is 54.5% and 70.2% for map tasks and reduce tasks, respectively. More importantly, these successful speculative copies improve the average task runtimes of reduce tasks (the average task runtime is decreased by 32.7%) and reduce the runtime of the slowest task by 48.7% (see Figure 7(b)). However, we can still see the natural skew-reduce issue: the gap between the longest reduce task and the average reduce task runtime is almost the same as in homogeneous environment (see Figure 4(b) and 7(b)).

It is clear that the reduction in the energy consumption is not proportional to the execution time reduction. This is due to the 32% increase (i.e., from 64.95 to 88.35 Watt as shown in Figure 6(b)) in the average power consumption (i.e., extra power consumption caused by speculative execution) and the significant decrease in the idle time, see Figure 9(b).

On the other hand, we can observe that in the case of Sort and WordCount, Hadoop cluster consumes more energy when enabling speculative execution. These results can be explained due to the increase in the execution time of the two applications and to the increase of average power consumption per node due to speculative execution. The average power consumption increases from 63.92 to 64 Watt when running Sort and from 80.7 to 81.68 Watt when running WordCount application.

It is important to mention that speculative execution successfully reduces the slowest map task by almost 80% in case of the Sort application, see Figure 7(a). But, due to the high number of speculative reduce tasks (all were not successful)
and the resulted network contention, we observe an increase in the slowest reduce task by almost 12%. As *Sort* is dominated by the completion of the last reduce tasks, this results in longer execution time.

In summary, we confirm that speculative execution — when necessary — can effectively mitigate stragglers, but may not necessarily reduce the overall running times of the applications. Moreover, we observe that the reduction in the energy consumption could only be achieved when the running time of the application is noticeably reduced. However, this reduction is not proportional to the performance improvement.

VI. MICROSCOPIC ANALYSIS

To complement our analysis, in this section we seek to understand the impact of speculative execution on the power consumption at node level. To do so, we first compare the average power consumption of a node in a homogeneous Hadoop cluster when varying the number of concurrent running tasks for the same application. Here we show the results when varying the number of concurrent reduce tasks for *CloudBurst*. As shown in Figure 10(a), our results indicate that the average power usage of a node gradually increases when increasing the number of concurrent tasks from 1 to 4 and it remains the same when the number of concurrent tasks is > 4. On the contrary, the average task runtime slightly increases when the number of concurrent tasks is ≤ 4. This gap is higher when the number of concurrent tasks is > 4. The same behavior is recorded while varying the number of concurrent map tasks of *WordCount* application (as shown in Figure 10(b)).

In summary, we find a clear trade-off between performance and power consumption when scheduling a speculative task. While launching speculative copies on idle nodes or nodes with a small number of running tasks result in lower average task runtime but leads to higher power consumption per node, launching speculative copies on nodes with larger number of running tasks result in higher average task runtime but lower power consumption per node.

Thus, as Section V-A suggested, speculative tasks allocation can significantly impact the energy impact of speculative execution. We now focus exclusively on *CloudBurst* and *WordCount* applications and study their sensitivity to speculative allocation.

We plot the CDF of the number of current running tasks on a node when launching speculative tasks. Figure 11(a) suggests that for the *CloudBurst* application, 62% of speculative tasks do not impact the average power consumption of the nodes on which they run (i.e., they are launched on a node hosting at least four other reduce tasks). However, the same observation does not apply to the *WordCount* application, as we can see in Figure 11(b), the power consumption of a node continues to increase when increasing the number of concurrent tasks per node. Thus, only 22% of speculative tasks do not impact the average power consumption of the nodes. This explains the results in Section V-A.

In summary, we conclude that an energy-aware speculative execution is necessary to reduce the energy consumption of Hadoop cluster. The energy-aware approach must consider the impact of launching speculative tasks on the overall energy consumption of Hadoop clusters. That is, to find where to schedule speculative copies in order to achieve the best trade-off between performance gain (i.e., reducing the slot occupation time of speculative copies) and power gain (i.e., minimizing the extra power consumption).

VII. DISCUSSION AND FUTURE WORK

In the era of Big Data and with the continuous growth of the cloud scale, energy consumption has become a challenging issue in recent years. Similarly, speculative execution has become of even higher importance in Hadoop systems when deployed on large-scale clouds where multiple diverse MapReduce applications share the same infrastructure. In this study, by the means of experimental evaluation, we have shown the impact of speculative execution on the energy consumption of Hadoop clusters. We have confirmed that straggler detection in Hadoop is not accurate and may lead to excessive unnecessary speculative execution and therefore increases the energy consumption of Hadoop clusters. We have quantified the energy cost of speculative execution:
we find that the average power consumption in the cluster when enabling speculative execution strongly depends on the duration of the speculative tasks (i.e., extra slot occupation), on the idle time, and on the allocation of speculative tasks. We conclude that speculative execution may result in a reduction in the energy consumption if and only if the running time of the application is noticeably reduced to compensate the energy cost of speculative execution. Finally, we discussed the trade-off between performance and power consumption when scheduling a speculative task.

Our future work lies in two aspects. First, to improve stragglers detection and handling in Hadoop: we plan to investigate a hierarchical detection approach that takes into consideration the root cause of slow tasks before declaring them as stragglers and design an energy-aware speculative execution that triggers speculative execution when energy reduction is expected. Second, we plan to design an energy-driven controller that targets improving the energy proportionality of Hadoop when enabling speculative execution: the proposed controller keeps track of the number of current running tasks per node and estimates their completion times. Consequently, these data are processed in order to determine the best energy and performance trade-off using different task allocation policies and leveraging energy-aware hardware configuration (i.e., putting node to sleep mode and using DVFS techniques). Our ultimate goal is to achieve the best performance with minimal extra power cost when using speculation and consider idle and sleep time to compensate the energy cost of speculative execution towards achieving energy proportionality in Hadoop.

ACKNOWLEDGMENT

This work has been supported by the H2020-MSCA-ITN-2014 project under grant agreement no. 642963. The experiments presented in this paper were carried out using the Grid’5000/ALADDIN-G5K experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see http://www.grid5000.fr/ for details).

REFERENCES