

Comments on the revised version

January 13, 2018

Let us first sincerely thank the reviewers for their relevant and in-depth remarks. Their comments helped us to clarify the focus of this work, which lead to restructure the evaluation procedure to some extent. As part of these modifications, we changed the title to: "Online Tuning of EASY-Backfilling using Queue Reordering Policies". It better reflects the main goal, which is not to advocate bandit strategies, but rather to study their feasibility, and to show where the tradeoff is compared to using simulations for tuning the algorithms. We hope this revised version may now be considered for publication. We detail our answers to reviewer's comments (bold).

1 Reviewer 1

The manuscript describes an experimental study to modify a scheduling approach common in many large parallel computer systems. The authors propose to replace the well-established First-Come-First-Serve approach combined with EASY-backfilling by a strategy using reordering of the queue. To prevent job starvation, the authors resort to FCFS for a job as soon as the waiting time of this job exceeds a threshold. Although this threshold approach is well known, the authors do not provide any references of previous publications using this approach.

The reviewer sees significant shortcomings in the experimental design of the manuscript. In general, computation experiments should follow the guidelines of experiments in natural sciences, that is, an experiment has the goal to prove or disprove a given hypothesis. The authors do not describe such hypothesis nor can the reviewer extract it from the manuscript (except for a trivial hypothesis: we can beat EASY-Backfilling for some traces).

Instead, the authors present a simple trial-and error study that exploits some parts of the problem space without justifying the restriction of the problem space. For instance, the authors suggest 12 reordering policies (Section 4.1) with the justification of the selecting a search space "as semantically diverse as possible" instead of carefully arguing which reordering policy has the potential to improve performance. For the scheduling objective, the authors use "the waiting times of jobs, which is one of the more commonly used objectives" although they state before that a "system administrator may use one or multiple cost metric(s)". Although this approach is the opposite of the approach used for the reordering policies, the authors do not justify the design decisions of their study.

General comments: Our scope in this work is is not to advocate particular scheduling strategies (Bandit), but rather to study their feasibility, and to show where the tradeoff is compared to using simulations for tuning the algorithms. We are aware that is it not so common in our field where the classical approach is to formalize an hypothesis and to assess it by experiments. The methodology that we propose here is to consider the 12 most popular (and efficient) reordering policies and to study their behavior. In the mean time, this also leads to figure out at the end if we can select the best ones. This methodology is closer to an empiristic study that are common in other fields. We added a brief justification at the end of the introduction. The reason why we have chosen this way was because it is very difficult to identify a priori what is the best policy and assess it. Our attempt was to consider all possible policies.

We answer below more specific points:

We focus in this paper on average waiting time. We have considered other objectives like bounded slow-down. This choice simply reflects the view of the users. Other metrics are indeed possible, and our method is agnostic to this choice.

We agree that thresholding jobs is a standard approach and added a reference that explores this aspect.

Similarly, the authors simply use 40h as threshold and simply state that the "choice of this default stems from the analysis developed in" a former manuscript of the authors that is so far unpublished. Such approach is simply unacceptable in view of the reviewer.

Right, this comes from a previous version, thanks for pointing this out. We completed the reference: "Tuning Backfilling Queues" JSSPP17.

The authors use 7 traces recorded over a time span of 18 years. They do not discuss whether these traces are comparable with respect to their study despite large differences in the number of processors, jobs and trace lengths.

In their algorithmic description, the authors include a decaying factor λ although they later simply state "this will not be studied in the experiments where $\lambda = 1$ and is left for future research". Therefore, the authors indicate that even in their own opinion, research regarding this algorithm is not complete but the presented findings are only a first step. The reviewer feels that such first step may be helpful in a workshop to discuss directions to complete such study but not for reporting the results of a study in a journal.

Right, we added an explanation for this in Section 6.1.3, page 8. This is due to the fact that the resampling technique generates stationary data.

Altogether, the reviewer believes that an experimental study with so many design problems is not relevant to the readers of TDPS. The reviewer does not see how the authors can overcome these shortcomings even in a major revision and therefore suggest rejection of the manuscript.

We disagree. We strongly believe that such a unusual approach is valuable, in the sense that it allows to get a sense of how useful simulators really are for the purpose of tuning backfilling heuristics.

In addition, the reviewer sees several minor problems in the manuscript that the authors can correct in a major revision. The reviewer only briefly describes such problems in the following:

Abstract: Any reader of TPDS can expect to understand the abstract without having to use additional literature. Therefore, the abstract should only include expressions that are common knowledge: it is at least doubtful that every reader understands the used abbreviations and expressions like "Epsilon-greedy multi armed bandit algorithm".

Right, we polished the abstract accordingly

Previous publications explain that studies using traces suffer from fill-ins in the starting period and from draining in the final period. The authors mention this problem but do not discuss their impact and strategies to avoid these problems although other publications present such strategies.

The use of our resampling method means the utilization is slightly lowered in the experiments comparing to the original traces. We do not believe this to be an issue, since we are studying adaptive methods in themselves, and not the underlying systems.

Previous studies have described that the KTH trace is hardly comparable with other traces due to a specific acceptance policy. The authors do not discuss this problem.

The traces we selected are quite disparate. This is on purpose to demonstrate the value of an adaptive approach: it works well on any kind of system. Moreover, our simulation model is simple: on real machines, we could reasonably expect the difference in behavior to increase. (See the answer to reviewer 3 on this point.)

The authors use either 1 week or 1 day feedback justifying their choices with the periodicity of the traces. Although such approach is reasonable, it is a hypothesis and requires validation by using other feedback periods as well and comparing the results.

We didn't set out to optimize the Δ hyperparameter. While we agree it could be optimized further similarly to ϵ by leave-one-out cross-validation, we don't believe it would add much to the discussion.

There are several publications discussing the disadvantages in using traces and suggesting workload models instead. The authors do not consider these studies in their manuscript without explaining why.

We leave this discussion to other works, in particular, we suggest the EuroPar article 'Workload Resampling for Performance Evaluation of Parallel Job Schedulers'.

Problems of the introduction explaining the goal of the study are likely related to the already mentioned design problems of the study. The same holds for the organization of the manuscript.

Right. We modified the introduction to better reflect the scope of the study. (See the last paragraph)

The authors can improve readability of the manuscript by explaining the underlying algorithmic approach in some more detail.

We added two figures to illustrate the algorithms in page 7 and 8.

Since the journal provides a platform for supplemental material, the reviewer does not understand why readers must contact the authors by email to obtain data to reproduce the study.

As previously concerning the missing reference, this was inadvertent on our part! This is fixed in the major revision. Thanks.

2 Reviewer 2

I basically liked this paper and support its publication. The following comments make some suggestions for possible improvements.

Thanks!

Title: I would suggest to remove the "(bandit)" from the title, and instead use something like "Online Queue-Ordering Policy Selection for EASY-Backfilling" – emphasize what you are doing, not how.

The need for more careful framing of the goals of the study was stressed by all reviewers. Accordingly, we made the scope of the study more clear and changed the title, to "Online Tuning of EASY-Backfilling using Queue Reordering Policies."

Related work: additional papers you may want to look at are

1. "Exploring portfolio scheduling for long-term execution of scientific workloads in IaaS clouds", from SC'13, suggests how to select from a large portfolio of possible schedulers;
2. "Improving and stabilizing parallel computer performance using adaptive backfilling", from IPDPS 2005, which also suggests online selection from a set of 4 alternative schedulers;
3. "Selective reservation Strategies for Backfill Job Scheduling", from JSSPP 2002, which suggests online decisions of whether to make reservations for skipped jobs - similar effect to your thresholding scheme;

4. "Self-tuning systems", from IEEE Software 1999, which is about simulating performance with local workloads to select systems parameter values.

Thank you for your suggestions. These papers are indeed very related to our study. They have been included in the references.

Section 2.2 - could be good to give a short description of how a basic version of such an algorithm works, instead of just noting a list of versions.

Right, we clarified this by adding a sentence describing principles of the algorithms.

Section 3.1 notation - intuitively p could represent processors just as well as processing time, so I would suggest t_i for time and p_i for processors. Similarly, in Section 4.1 I would find the names more intuitive with SJF/LJF instead of SPF/LPF (shortest/longest job first) SPF/LPF instead of SQF/LQF (smallest/largest processors first) expansion factor is often called slowdown area is often called total resources

Well, our notations are coherent within the whole paper. We agree that p_i may be confusing for some readers. However, since we lack of a standard, we chose to use the classical notation of scheduling theory. Regarding the way to name the policies, they are different acronyms used in the literature, again we chose to be consistent with the p_i and q_i notations - SPF corresponds to the smallest p_i first and SQF refers to the smallest q_i first. We added the remark concerning the total resources and the expansion factor. This notation comes from the "Characterization of backfilling strategies for parallel job scheduling" paper, where the authors call *Expansion Factor* the value of the slowdown where the upper bound on the running-time is used.

Section 5.1 exact simulation: I understand why you divide the time into periods as part of online tuning. But I disagree with doing so also in the simulations - as you yourselves say, it just causes problems with boundary effects. And you can discount more distant jobs without using periods.

We agree that this design choice can be discussed. Our decision to divide makes the study more realistic with respect to the cases where simulations are expensive (which is the case for most large systems). Indeed, the more realistic simulations, the larger time they take. For example, precise simulation frameworks such as SimGrid/Batsim have large runtimes, and simulating the history of the whole system every period is too prohibitive if the network topology and/or multiple resource constraints are used. As for discounting - it can indeed be used in both cases.

Section 5.2 - adding noise to the output rather than to the input workload doesn't make sense in my opinion. And the result that it gave the same results is totally expected - you add random noise with mean 0 and then find that the average didn't change.

We agree that this does not change the estimate - the point of adding this noise was to show that the sample size is not prohibitive in practice. Indeed, we learn the policy almost as fast with this noise, which motivates the use of the technique in real-world cases. The main motivation for introducing this noise was really to compare the difference in the shapes of the two green curves in Figure 3.

Section 6.1.1 traces - I think negative values are just -1 meaning no data is available.

You are right, we clarified this point in the text (Section 6.6.1 - Page 7).

Section 6.3.1 - I think you can say more about the post-factum results. Even if it is not your focus, it is still interesting that for example ordering by size is more beneficial than by time, and that it should be largest-first (and the SJF-like scheme is not good).

It is an interesting point, which was addressed in our previous publication 'Tuning Backfilling Queues' (JSSPP17). Here, we prefer to be cautious. Indeed, the high number of simulations in this work forced us to use a simplistic simulator model. We can reasonably expect these results to change when adding a topology model, or a model of job interference for example. While we believe this simple model is sufficient to study the behavior of adaptive policies, we prefer not to draw conclusions about the systems themselves from these simulations. We added this remark in the text (Section 6.3.1 - page 10).

Section 6.3.2 FCFS vs. Random: you justify the result be a conjecture that jobs do not pile up in the queue. You should verify this by counting the jobs in the queue in the simulation.

We agree that this conjecture is not warranted, and thus, we removed it in the revision. We have plans to explore this aspect in a future work where the discussion would be more appropriate.

Table 3: add columns with best static result for comparison.

Done, thanks you for the suggestion.

Description of Figure 6 results: this is not slow convergence, it is no convergence. and using $\epsilon=0.5$ is much too high: it means that half of the time you are choosing randomly, and that's why all of the possible approaches are used approximately the same fraction of the time. What you want is to initially use them all, but then reduce ϵ with time till you reach some reasonable value like random exploration in 1 of 10 or 1 of 20 periods.

We changed the experimental protocol to use leave-one-out cross validation, which did improve the bandit results. The selected exploration rate was 0.1 for all traces, although it is not the best choice for all of them. (see the new Table 2) The choice of a fixed exploration rate is one of simplicity - while decaying exploration does provide better performance, it is also harder to tune.

Reference [4] is often referred to and gives important background, but is not available. **NOTE: this journal does not really do blind review. Your names are given on the first page.**

Right, this is corrected in the major revision. The reference is 'Tuning Backfilling Queues' (JSSPP17) and it has been added in the references.

Typos and usage [...]

We incorporated all the suggested changes, thanks.

3 Reviewer 3

I think that the current experiments are not sufficient to show the advantage of the proposed strategy. The authors compare the performance of the proposed strategy with the baseline performance derived by EASY Backfilling. The performance of EASY Backfilling is good metric as the baseline, but comparison with EASY Backfilling is not enough to discuss the advantage of the proposed strategy. A lot of work about tuned Backfilling has been presented as discussed in Sec. 2. The author should discuss the performance improvement compared with the existing tuned backfilling strategies. The experimental results in Table 2 show that LAF and LQF show best performance among 12 reordering policies. This indicates that we can expect significant performance improvement by using static strategies, Backfilling with LAF or that with LQF, compared with EASY Backfilling. If the performance gap between the proposed strategy and the static strategies are small, the static strategies may be preferable because they are less complicated and run with smaller scheduling overhead. A comparison between the proposed strategy and the static strategies is needed to show the advantage of the proposed strategy.

Right, we agree that in order to really motivate the use of bandit/resimulation approaches, one should compare the performance of static policies, say using leave-one-trace-out cross validation, to the performance of the adaptive variants, using leave-one-trace-out cross validation for their hyperparameters. However, this work does not aim to emphasize the necessity of the bandit approach (the title has been changed and the introduction has been modified in the revision to better reflect this). Our goal was rather to show its advantages, feasibility and limitations. There are two main reasons for not trying to be too demonstrative of the superiority of the adaptive approach using our hypotheses, namely:

(1) First, this study used a limited amount of workload logs. In consequence, the process of comparing the adaptive (hyperparameter-cross-validated) choice to the static (cross-validated in the set of fixed policies) choice would not be really conclusive. Indeed, the more the number of workload logs, the higher the diversity and the more adaptations are needed.

(2) Second, we need to run high number of simulations, and accordingly we used a light simulation model . In this model, we do make machines *more* similar to each other by neglecting topological differences. It can be reasonably expected that differences between systems should *increase* rather than decrease, when moving away from a simplistic model to the real systems. This reason is also why we don't expand on which fixed strategy is the best. Indeed, the results are probably different when adding a topology model, job interference, or an I/O congestion model. We do believe this simple model is enough for studying adaptive policies though. We added a remark in the text (Section 6.3.1 - Page 10)

We modified the present experiments to use leave-one-out cross validation in the hyperparameter selection, and we clarified the reasons for choosing this experimental protocol. We are preparing a subsequent study using an approach closer to what you describe, in which we greatly increase the number of workload logs used.

The authors assume +/-20 % noise in the simulation running in the proposed strategy. However, the reason why this assumption is suitable is not presented. The authors should discuss the noise of 20 % is suitable to represent real world jobs.

Right, a justification was clearly missing, this value is taken as to be sufficiently larger than the values previously reported, see e.g. "Batsim: a Realistic Language-Independent Resources and Jobs Management Systems Simulator" which reports few percents of variation of the simulated metric compared to the metric on the real systems. We added this remark in the text (Section 5.2- Page 6).

Minor comments for improving presentation

Thanks, we incorporated all the corresponding changes in the text.