Comparing Performance Heatmaps

David Krakov and Dror G. Feitelson(✉)

School of Computer Science and Engineering,
The Hebrew University of Jerusalem,
91904 Jerusalem, Israel
feit@cs.huji.ac.il

Abstract. The performance of parallel job schedulers is often expressed as an average metric value (e.g. response time) for a given average load. An alternative is to acknowledge the wide variability that exists in real systems, and use a heatmap that portrays the distribution of jobs across the performance $\times$ load space. Such heatmaps expose a wealth of details regarding the conditions that occurred in production use or during a simulation. However, heatmaps are a visual tool, lending itself to high-resolution analysis of a single system but not conducive for a direct comparison between different schedulers or environments. We propose a number of techniques that allow to compare heatmaps. The first two treat the heatmaps as images, and focus on the differences between them. Two other techniques are based on tracking how specific jobs fare under the compared scenarios, and drawing underlying trends. This enables a detailed analysis of how different schedulers affect the workload, and what leads to the observed average results.

1 Introduction

Graphs and visualizations are generally acknowledged to be important tools for understanding complex systems. But as Anscombe [1] noted, few of us escape the notion that graphs are rough, and when dealing with large amounts of data numerical computations are the "real" analysis. Such analysis assumes that we are able to find a numerical formula that is congenial to the data. However, numerical recipes typically entail an abstraction and simplification, while in many real-world examples it is best to first look at the data in all its complexity and observe its behavior, perhaps forgoing more formal analysis altogether [2].

Parallel job scheduler performance evaluation is no exception to this rule. The conventional approach for performance evaluation of parallel jobs schedulers tries to condense the varied information about scheduling behavior into few representative metrics such as the *mean response time* (the time between job submittal and completion) or the *mean slowdown* (the ratio between the actual runtime and the response time, which includes the wait time). However, using the mean fails miserably on asymmetrical distributions [4]. Slowdown is especially problematic, as short jobs may have disproportionately high slowdowns. Downey [5] calls the tendency to report summary statistics for non-symmetric distributions that exist in real workload data a "bad habit"; Frachtenberg [7] mentions using the mean for asymmetrically distributed (skewed) results as one of the pitfalls of parallel job scheduling evaluation.

A major reason for using simple condensed metrics is the need to compare different job scheduling strategies applied to the same workload, or to show how performance depends on some parameter (e.g. how simulation results change with load conditions). But if our metrics (and specifically, their averages) mis-represent reality, any comparison based on them becomes questionable. As research of parallel job scheduling strategies commonly revolves around comparison of alternatives, finding a good comparison tool becomes an important problem.

Heatmaps are a powerful tool to visualize large amounts of data, and are gaining hold in various areas as an analysis tool for complex information. When applied to parallel job schedulers performance evaluations they can help to visualize how various job metrics distribute relative to different load conditions [8], in contrast to the traditional approach of using the average metric as a single data point for evaluation. Thus heatmaps can help avoid the pitfall of misleading performance metrics.

However, it is difficult to visually compare different runs. Such comparisons are important to identify odd patterns and find better behavior. But — as the examples we explore will show — heatmaps may "look the same" despite having important but subtle differences. Naturally, since heatmaps visualize a job distribution over the load $\times$ performance space (as recorded from some scheduler and workload), one can use statistical analysis tools to compare different job distributions and obtain numerical results. But as we will show, this approach can be misleading as well.

As an alternative, we suggest to stay in the visual domain. We therefore need ways to highlight changes in behavior and allow high resolution analysis of the differences between heatmaps representing different scheduling algorithms or conditions. We present a number of such techniques in this paper. The first is simple image subtraction. The second is based on ratios, and is similar in spirit to the Kullback-Leibler divergence. Two additional schemes are based on tracking how individual jobs move across the load $\times$ performance space.

2 Heatmaps Applied to Evaluation of Parallel Job Schedulers

The performance of a computer system obviously depends on the workload it handles. Reliable performance evaluations therefore require the use of representative workloads. The workload used for the evaluation should represent real job distributions, including internal correlations and structure emerging from user feedback and system boundaries. As a result, evaluations of new schedulers often use workload logs obtained from real production systems to drive a simulation. Many real world traces are available at the Parallel Workloads Archive [11], converted into the Standard Workload Format (SWF) [3].

Heatmaps were recently proposed as a tool that allows high resolution analysis of a job trace or of simulation results [8]. This is based on the observation that a single trace or simulation contains a wealth of information about behavior under different load conditions, which can be exploited to zoom in on different conditions [12].

Specifically, a heatmap is like a scatter-plot showing the distribution of the jobs in the log. In our heatmaps the X axis is the load experienced by each job, and the Y axis is the job's performance, as measured per-job by common metrics such as slowdown or response time. The heatmap image is based on a fine grid where each bin shows how many jobs experienced the load and performance represented by the bin's coordinates. "Hot" (dark) bins mean lots of jobs and "cold" (light) bins are few jobs.

We use the same engine for heatmap calculation as was used in our previous work [8], and have made its code available online [13]. The load experienced by a job is computed as the weighted average of the utilizations of the system during the job's lifetime. During a job's lifetime the utilization changes at discrete points when some other job either starts or terminates. Assume this happens n times, and denote the job's arrival time by t_0, its termination by t_n, and the utilization at interval i by $U(t_i, ti+1)$. The load experienced by the job is then

$$\text{load} = \sum_{i=1}^{n} \frac{t_i - t_{i-1}}{t_n - t_0} \, U(t_{i-1}, t_i)$$

Since the distribution of most metrics is very skewed (for example for wait time, there are very few jobs that wait a lot and many jobs that wait a little, covering times from few seconds to days), the Y axis is plotted in logarithmic scale. Bin shades are also log scaled — there are many bins with very few jobs (typically 1–2) and few bins with a lot of jobs (hundreds). The bins are hexagonal and use a 50×50 grid with approximately 2,500 bins.

As an example, consider the behavior of the job scheduler of the CTC-SP2 system based on a year-long trace. Figure 1 shows a heatmap of the wait time jobs experienced in the original trace as a function of the load. The blue X marks the spot of the average wait time and load. Few observations are evident:

1. The mean wait time is not very representative of the distribution.
2. There is a distinct blob of jobs at the left side that seems to reflect a set of jobs that suffered from some congestion condition.
3. Wait times follow a bimodal distribution, with many jobs concentrated around low wait times and many more around a higher wait time. The short wait times may reflect some minimal granularity of activating the scheduler, meaning that it only runs say once a minute and therefore does not schedule newly arrived jobs immediately.
4. There are many jobs in the "background", showing no distinct relation between load level and experienced wait time.

Moreover, compare this with an EASY simulation based on same job arrivals as the original trace, shown in Fig. 2. The simulation is quite different from

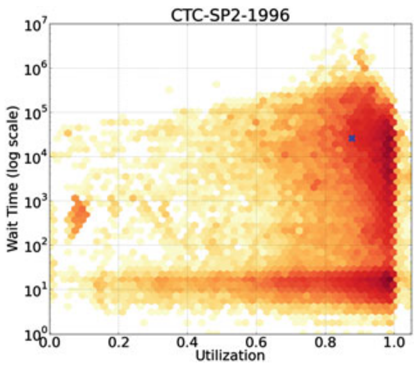


Fig. 1. Heatmap of the original log from the CTC-SP2 computer, showing asymmetrical job distribution.

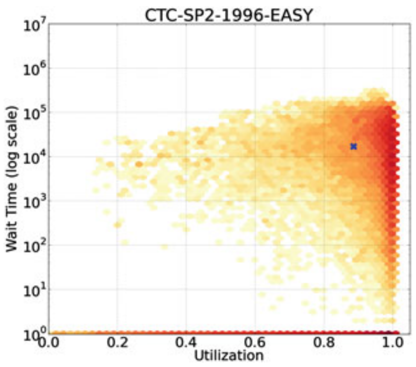


Fig. 2. Heatmap of EASY simulation based on the CTC-SP2 log.

the original log, notably in the spread of jobs and in the maximal wait times observed. Generally, looking into heatmaps of many real world systems, it has been shown that both real world traces and simulation results exhibit distributions for which the traditional mean-based comparison metrics are ill-suited [8].

For our comparison evaluation we use FCFS, EASY backfilling, and conservative backfilling (CONS) schedulers. In FCFS (First-Come-First-Serve) jobs are kept in order of arrival and whenever there are enough resources (free processors) for the first queued job, it is allocated its required resources and starts to run. If sufficient resources are not available, jobs are queued waiting for them to become available. Jobs never jump each other in the queue.

EASY [9] is a popular backfilling algorithm, and the most commonly used method for batch scheduling [6]. It approaches the problem of idle resources in the FCFS setting using the following optimization: when a job arrives or terminates the scheduler scans the queue of waiting jobs by order of arrival, running them as available processors allow. Once it reaches a job it can't run (requires more processors than available) the scheduler makes a *reservation* for the job. A reservation guarantees enough processors at the earliest time when the job will be able to run, based on current knowledge of when enough processors will become available after termination of currently running jobs. The scheduler then continues to scan the queue for smaller jobs that can be started without interfering with the reservation, and executes them. This action is called *backfilling.* Since the scheduler can not know when the jobs will end, it relies on run time estimates provided by the user. Once a running backfilled jobs exceed its user estimation it is killed by the scheduler to ensure waiting jobs with reservations start on time. "EASY_P", a hypothetical EASY with "perfect" exact user estimations, is also used as a reference for comparison.

CONS [10] is the "vanilla" backfilling algorithm, in which no job can delay any previous job in the queue. All jobs in the queue receive reservations upon their arrival (unlike the more aggressive EASY which reserves only the top job on the queue). Thus, response times and order are guaranteed upon submittal.

3 Heatmap Comparison Techniques

Heatmaps serve as a tool to investigate the whole population of jobs instead of summary statistics such as the average slowdown. Thus, by comparing the heatmaps describing the performance of a certain log under two different schedulers, we may hope to achieve a fine-grained comparison of these schedulers. However, no common practice exists for the comparison of different heatmaps. We suggest two possible approaches:

- **Visual heatmap comparison:** Visual (image) comparison can help identify changed areas between two job distributions. Since it is image based, such a visual approach can be used to compare vastly different populations. The input of this approach is a set of two heatmaps (two matrices of same size), and the output is an image that highlights differences between these heatmaps. Uses include (1) comparison of different schedulers on the same workload, especially powerful for comparing similar schedulers to highlight small changes in the resulting performance distribution; and (2) comparison of scheduling of different job sets, and observation of the behavior of the same scheduler over different workloads.
- **Job aware comparison:** If the compared heatmaps portray performance distribution of the same workload, that is the same set of jobs but under different schedulers, it is interesting to compare the performance experienced by individual jobs. Visual comparisons as described above do not use this information. The aim of this approach is to identify trends in the behavior of specific classes of jobs, such as what happens to high slowdown jobs, or how jobs with low run time are affected. This approach does not directly compare heatmaps — instead it shows how sets of jobs that created one heatmap moved in order to create the second heatmap.

The next two sections detail these two approaches.

4 Visual Heatmap Comparison

The visual approach is completely general, and can be applied to any pair of heatmaps. In its simplicity and applicability lies its greatest advantage.

A naive image comparison is employed. Each heatmap is actually a rendering of a 2D histogram, showing the number of jobs in each bin. The bins in the heatmaps correspond to each other, with a bin for each combination of load and performance. Denote bin i in the two heatmaps by A_i and B_i. Then, calculate the difference between the values for each bin, and determine the color based on whether is it less or greater than zero. We use two different differences:

Simple difference $D_i = B_i - A_i$

Ratio difference $D_i = \frac{B_i}{A_i}$

In both cases, we assign colors to differences using a logarithmic scale. Thus successively darker shades indicate growth in the order of magnitude of the difference. This approach helps to identify subtle differences between possibly very similar distributions of results. For example, see Fig. 3, which compares two very similar runs of the EASY scheduler. The second simulation (with somewhat better backfilling due to using perfect runtime estimates) has less jobs with high wait times across all utilization levels. The visual produces even more pronounced results when simulations are different enough — for example, Fig. 4 shows unsurprisingly clear advantage of EASY over FCFS.

In the above examples we use red and blue shading to distinguish whether the first or second heatmap dominates. This shows up prominently in color displays, but is not suitable for black and white printing. In a black and white version, equality can be gray, with advantages for one heatmap shown in darker shades and advantages for the other shown in lighter shades.

Figure 5 shows side-by-side comparisons between simple and ratio differences. Ratio differences sometimes create wider boundaries between areas where either heatmap dominates — as the SDSC-SP2 EASY vs. EASY_P comparison. This happens because in the boundary areas the advantage of one heatmap over the other is usually small in relative terms, even if it is large in absolute terms. On the other hand ratios are more susceptible to noise, as the noisy SDSC-SP2 CONS vs EASY ratio comparison shows, while simple difference clearly shows the advantage of EASY over conservative backfilling for that workload.

Relation to Kullback-Leibler Divergence

A comparison of two heatmaps is essentially just a comparison of two histograms, or distributions. Thus an alternative to our visual approach is to use statistical tools that compute the difference between two distributions. Applying this to our context, we can compute the difference between how two sets of jobs distribute over the performance × load space.

One commonly used tool to measure difference between distributions is the Kullback-Leibler divergence D_{KL}. This comes from information theory, and is typically used to compare a model distribution with the underlying "true" theoretical distribution. Specifically, given a base distribution P and an approximation Q, the divergence is defined as

$$D_{KL} = \sum_{i=1}^{n} P(i) log \frac{P(i)}{Q(i)}$$

If we're just interested in comparing distributions, this expression can be interpreted as follows. First, for each possible value i, find the ratio of the probabilities to observe i under P and Q. Then take the log of this ratio, to find its order of magnitude. Next, weight these values by $P(i)$, meaning that more weight is assigned to the more probable values. Finally sum it all up to obtain a measure

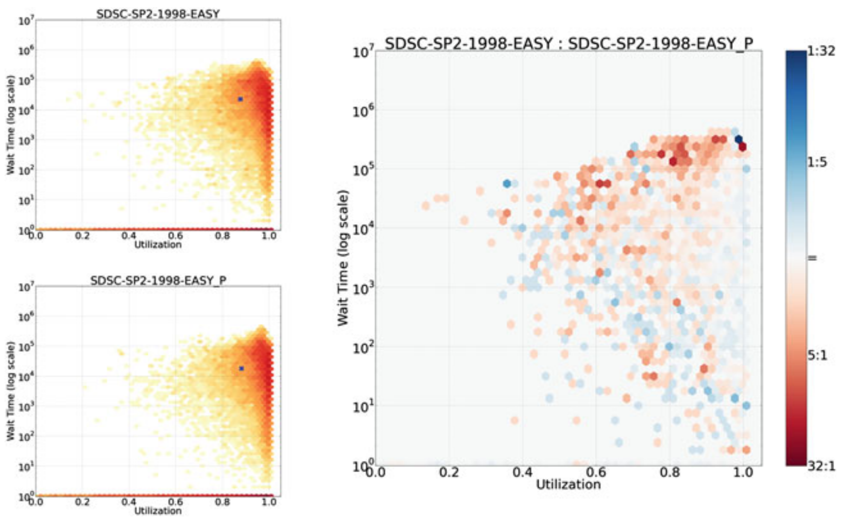


Fig. 3. Ratio comparison of two simulations based on the SDSC SP2 workload: EASY and "perfect" EASY (EASY_P). The original heatmaps are very similar. The darkest blue indicates that EASY_P had 32 times as many jobs in this bin as EASY (Color figure online).

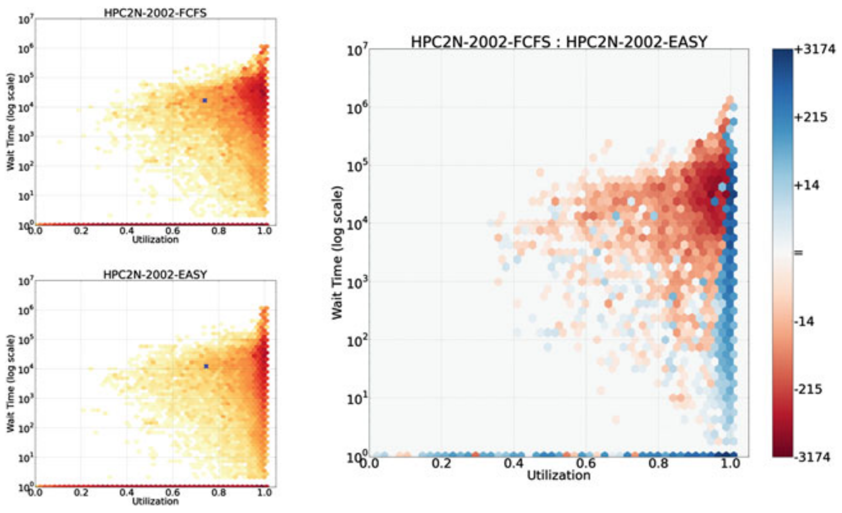


Fig. 4. Simple difference comparison of two different simulations (FCFS and EASY) based on the HPC2N workload. The darkest blue indicates that EASY had 3174 more jobs in this bin than FCFS (Color figure online).

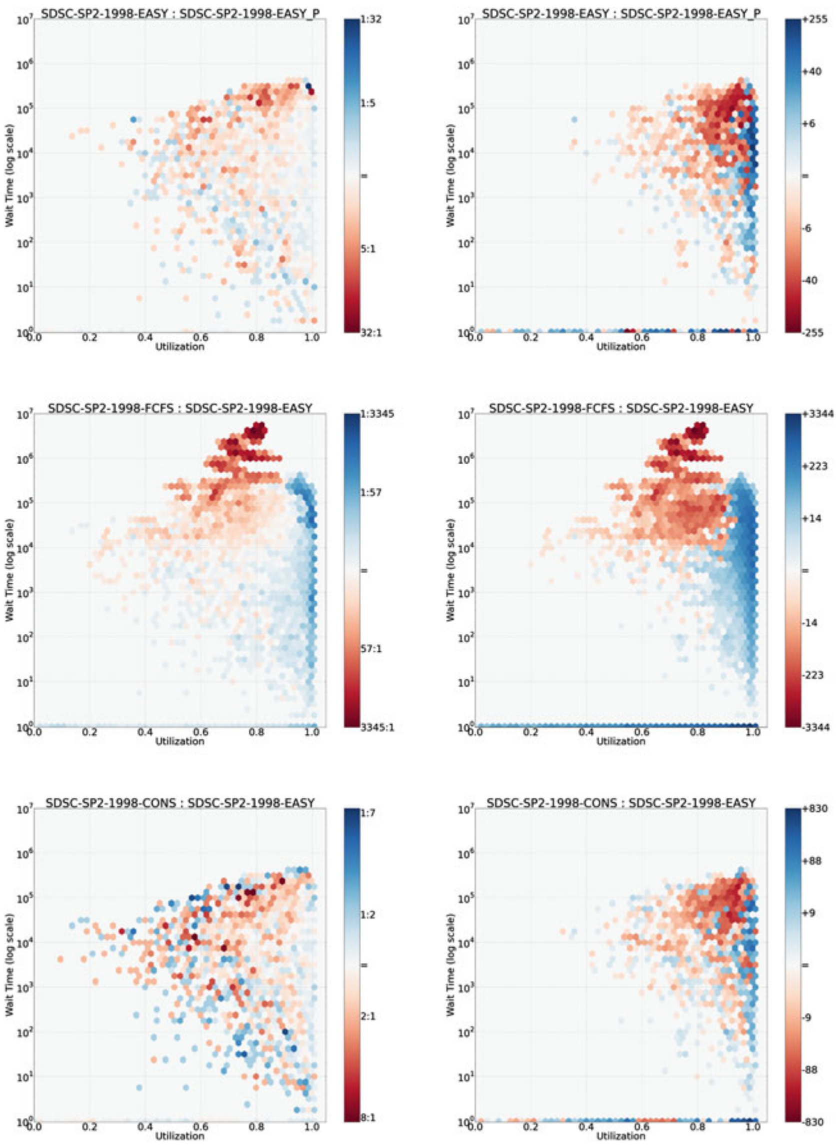


Fig. 5. Comparison of ratio differences (on the left) and simple differences (on the right) for the same pairs of heatmaps. Ratios show that some of the large absolute differences are actually small in relative terms.

of the overall divergence. A small divergence signifies similar distributions, or a good model. Note that the first steps are the same as in our ratio difference: we assign colors based on the log of the ratio.

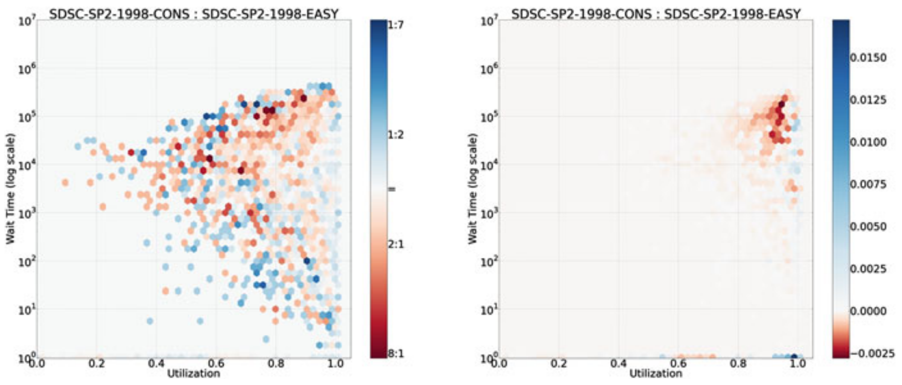


Fig. 6. Left: ratio comparison of between EASY and conservative backfilling for SDSC-SP2. Right: weighted version of same results (D_{KL} inspired). Note color levels adjusted to accommodate asymmetry around zero level. Note that most of the high ratios apparently have low weights.

To apply this to our performance heatmaps, we can transform each bin into a probability by dividing by the total number of jobs: $P(i) = j_i/N$, where j_i is the number of jobs in bin i. But due to the ratio, D_{KL} is only defined if $P(i) = 0$ whenever $Q(i) = 0$ (absolute continuity). The simplest practical approach to ensure this uses a uniform prior, which amounts to adding 1 to each bin and renormalizing the heatmap. Denoting the number of bins by S we then get $P(i) = (j_i + 1)/(N + S)$, and similarly for Q. These are plugged into the equation to compute D_{KL}.

When comparing performance heatmaps, D_{KL} has two shortcomings. First, it quantifies only the magnitude of the difference, with no indication which is better. Second, due to summing over all possible values, different pairs of distributions may lead to similar results. In particular, things may cancel out, leading to situations where a meaningful difference receives a very low divergence score.

Consider as an example Fig. 6 comparing EASY scheduling based on SDSC-SP2 workload with conservative scheduling (CONS). The computed D_{KL} is particularly low (0.084), indicating no significant difference. But looking at the weighted ratio heatmap we find that there are in fact two areas with relatively high probability and different patterns for the two schedulers, that apparently cancel out:

- With CONS the odds are higher than EASY for jobs experiencing higher wait time and slightly lower load. This combination results from using more reservations, more waiting in queue, and thus overall lower utilization.
- With CONS there is considerably less chance of a job to experience no slowdown with full utilization. One possible explanation is that these bins may represent small jobs that are executed under EASY due to aggressive backfilling.

5 Job-Aware Heatmap Comparison

Job aware comparisons use heatmaps as a tool to map what happened to certain classes of jobs by creating a heatmap of those jobs only and trying to generate a "before" and "after" view. It relies on domain specific understanding and especially the fact that the compared heatmaps show distributions of the same jobs. Thus this can't be used to compare arbitrary heatmaps or even the results of simulations using different logs.

Two approaches are explored: per-area comparison, where we plot heatmaps of jobs from select areas only in the original heatmap, and quiver plots, where we plot the trends of all jobs by creating a "stream flow" plot.

5.1 Area Plots

To compare performance results A and B for the same workload, we split the heatmap of A into a mesh sized $N \times M$, and plot a separate heatmap based on B for jobs that come from every square of the A mesh.

We use a 3×3 mesh in the figures below. Thus we partition the A heatmap into 9 squares, and partition the jobs in the workload into 9 sets, such that the jobs in each set contributed to one square of the heatmap. The sets correspond to all possible combinations of low, medium, and high load with low, medium, and high performance. In other words, we get 9 sets of jobs based on how they fared in the A evaluation: those jobs that experienced high load and low wait times will be in one set, those that experienced high load and medium wait times in a second set, those that experienced high loads and high wait times in a third set, and so on. Then we use the second evaluation to draw a separate heatmap showing the load and performance of the jobs in each of these sets.

Note that the sets need not be of equal size. Usually there are very few jobs that enjoy a low load but still suffer from high wait time or slowdown values, and sometimes there may even be no such jobs. On the other hand there are many jobs that experienced high loads, both with high and low performance.

The advantages of this approach are that it allows to keep the high resolution analysis (a heatmap) for classes of jobs. It can easily be extended to plotting any groups of jobs (such as jobs by a specific user) instead of basing classes on performance areas in A.

Results. When comparing different simulations using the same workload data set, the area plots allow to see differences in behavior:

– Figure 7 shows EASY vs. EASY_P ("perfect EASY", same as EASY but uses actual job runtimes instead of user estimates to determine reservations). Perfect EASY clearly performs better — all jobs with high wait times experienced lower wait times and moved "down" on the plot. Surprisingly, the class of few jobs with low wait times and high utilization (lower right) saw a substantial increase in wait times. Better packing comes at the expense of the jobs that did not previously wait in high load conditions.

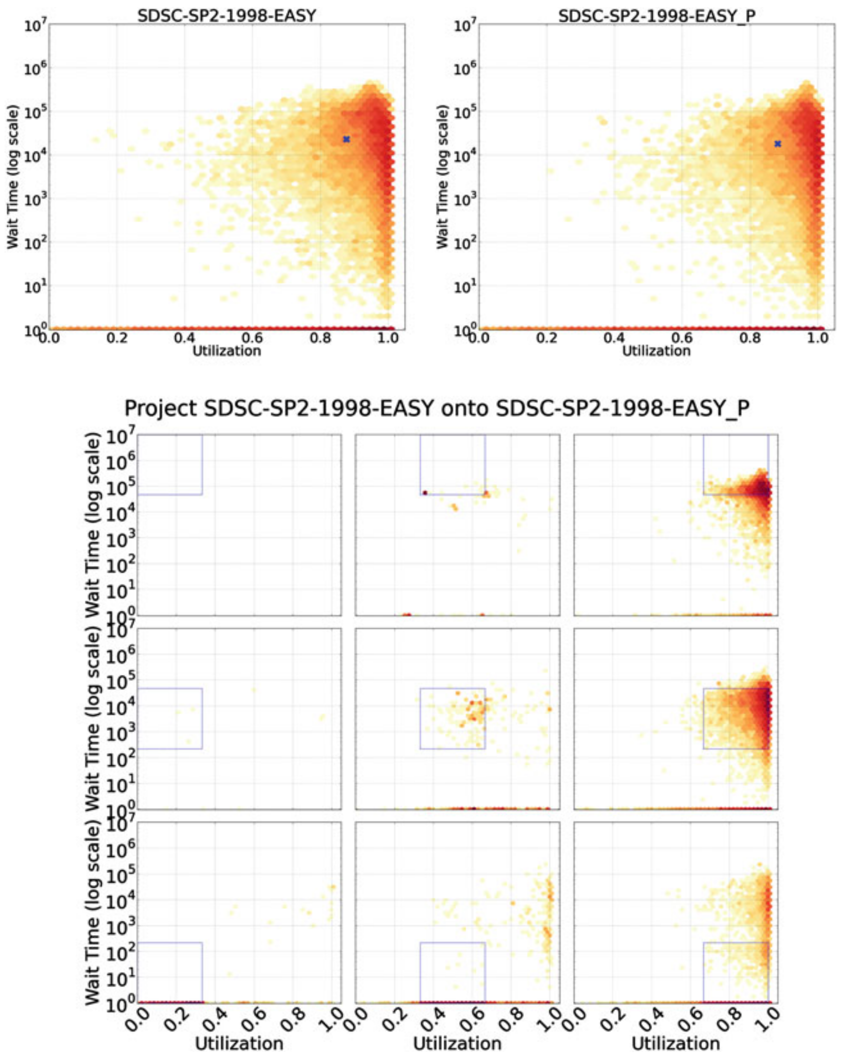
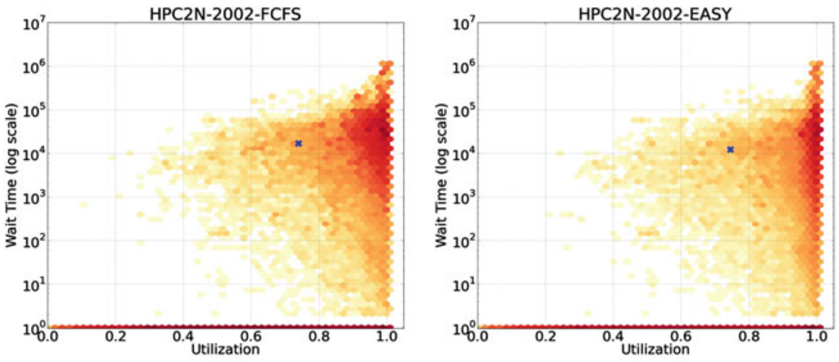


Fig. 7. Comparison of SDSC SP2 EASY and EASY_P simulations. The heatmaps derived from the two simulations are shown on top. The array of 9 heatmaps below show disjoint areas in the EASY heatmap (blue outline), and how the jobs in these areas were mapped to the EASY_P heatmap (Color figure online).

– Figure 8 shows FCFS vs. EASY, based on the HPC2N workload, which produced rather similar plots. Jobs in upper right were strongly affected, experiencing much lower wait times. Jobs in the center and middle right were affected as well: many jobs moved to the bottom zero wait-time line, probably because other jobs were packed better and left more processors free.



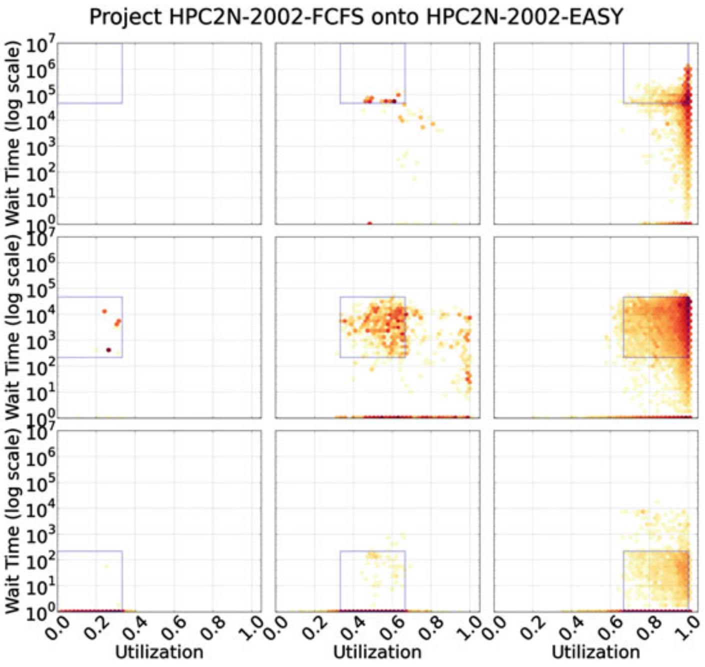


Fig. 8. Comparison of HPC2N FCFS and EASY simulations.

On the other hand, comparison of original schedulers versus simulations (Fig. 9) shows that there was very little relation between the job distribution in the original log and what these jobs experienced in the simulation. No matter where the jobs were originally, the distribution of each subset in the simulation is essentially the same.

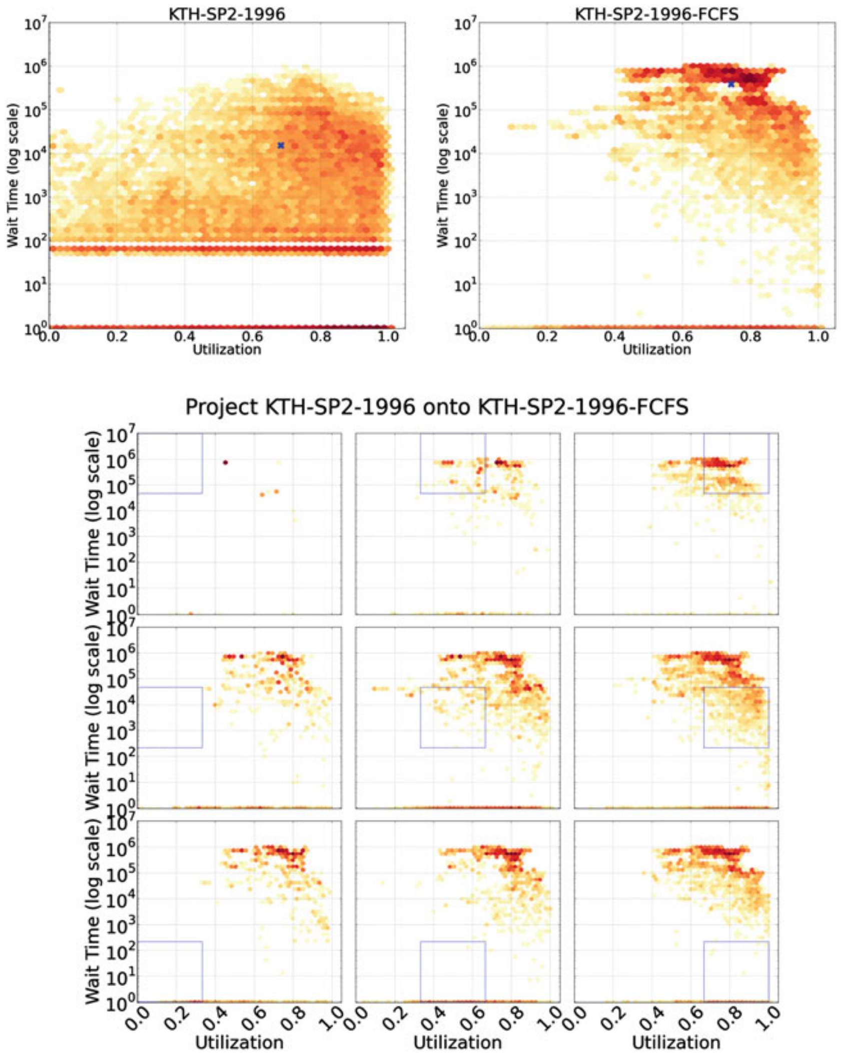


Fig. 9. Comparison of original KTH workload and FCFS simulation.

5.2 Quiver Plots

In this approach we split the A heatmap into a grid of relatively high resolution (a 40×40 grid with 1,600 cells is used). For each cell we potentially draw a quiver as follows:

1. Let J be the set of jobs in that cell of A. Look at the subset of jobs $J_{out} \subset J$ that leave the cell, that is that either the load they experienced (their x coordinate) or their performance level (their y coordinate) are out of the cell boundary in the B heatmap.

2. A quiver (vector) is plotted only for cells with more that a minimal number of leaving jobs ($|J_{out}| > 25$). This is done to reduce the amount of "noise" from changes in insignificant cells. The minimal number (25) is tuned for the logs analyzed, but an adaptive approach based on the average cell size can be employed.
3. Compute the placement of the quiver. The quiver's tail is placed at the mean location of the jobs in the A heatmap (by definition this is within the cell). Its head is placed at the mean location of the J_{out} jobs in the B heatmap:

$$tail.d = \frac{1}{|J_{out}|} \sum_{j \in J_{out}} j^A.d \quad \text{for} \quad d \in \{x, y\}$$

$$head.d = \frac{1}{|J_{out}|} \sum_{j \in J_{out}} j^B.d \quad \text{for} \quad d \in \{x, y\}$$

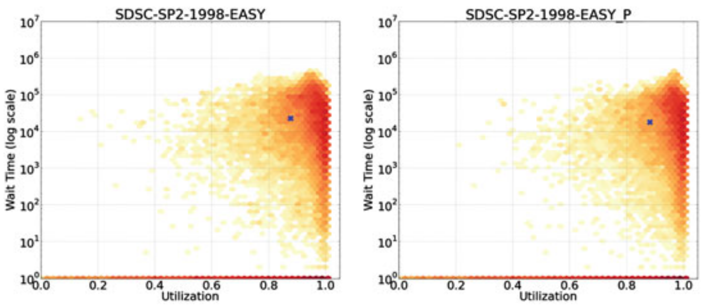
4. The quivers are based on only the leaving jobs to allow for a visually meaningful vector even when only a small percentage of the jobs are leaving, and so the mean of *all* the jobs is close to the cell or even within the cell. The color of the quiver is based on the percentage of the jobs moved, making it easy to distinguish cases where only a small fraction of the jobs moved from cases where most of the jobs moved.

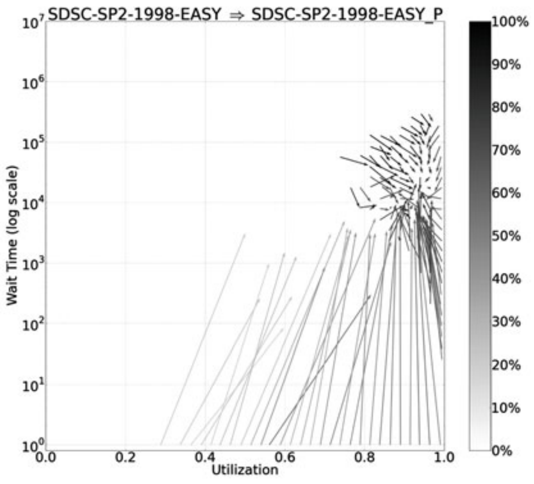
Results. We first look at behavior of similar schedulers to check if trends can be identified, and whether there is symmetry between them. Consider the comparison EASY and EASY_P based on the SDSC SP2 data trace, Fig. 10.

- Top, EASY → EASY_P: jobs move towards a new "center" with higher load and lower wait time. Some percentage of the jobs with low wait time along the high load 1.0 axis may experience higher wait times in result. Jobs that experienced no wait time at all load levels may experience high wait times — the "picket fence" on the bottom. Zooming in on bottom cells shows a very uneven distribution: most of the jobs move right or left in terms of experienced load, but stays at zero wait time, while a small group (up to 40 %) "enters the game" and experiences higher (10^3) wait time closer to the average.
- Bottom, reversed plot of the same two simulations. A new center of weight draws jobs to it, with higher wait time and lower load. Interestingly, two trends are very similar to the previous plot. First, the "picket fence" at the bottom is present in both plots, as there are distinct jobs that had to wait in one scheme but not in the other. Second, Jobs that had an almost perfect packing (100 % experienced load) and low wait time move to the left, now experiencing higher delay. It seems that for the EASY scheduler both these effects are common to any run.

A similar center of weight shift can be seen when comparing EASY vs. Conservative backfilling, based on a number of different workload traces (Fig. 11).

When comparing the non backfilling FCFS to EASY (HPC2N workload based simulation in Fig. 12), EASY clearly helps lower wait times of jobs in all load levels and create a denser packing, resulting in higher achieved load. There are





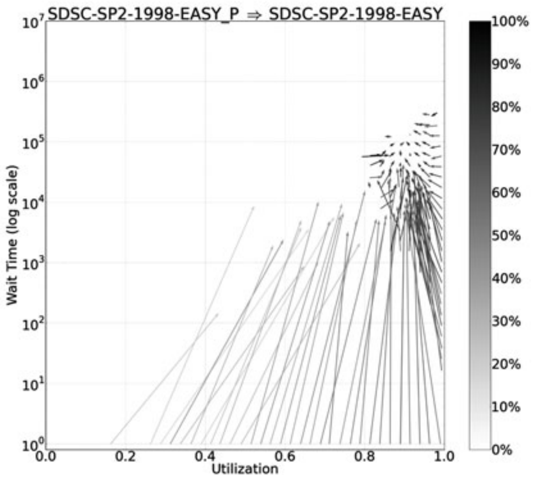


Fig. 10. Quiver plot comparison of SDSC SP2 EASY and EASY_P simulations. On the top the changes in job performance when moving from EASY to EASY_P, and on the bottom the other way around.

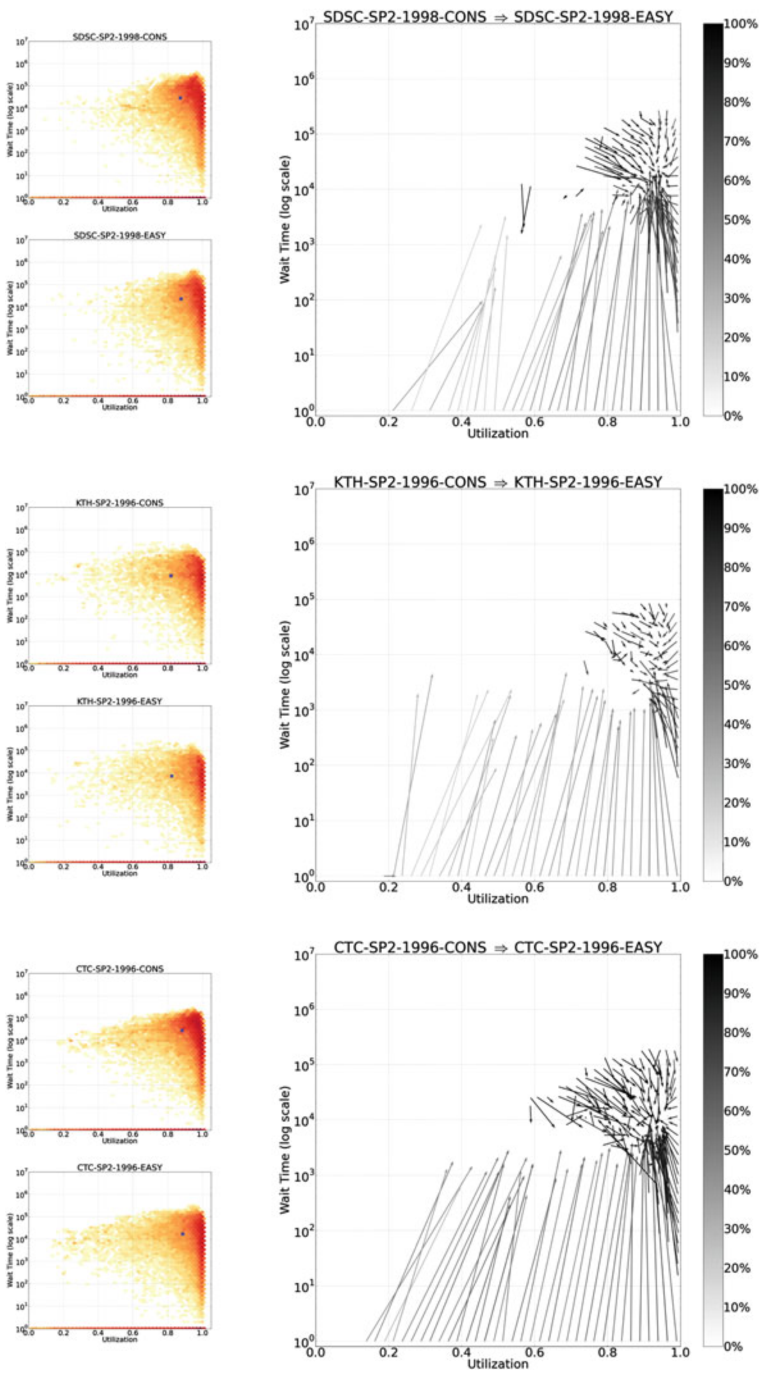


Fig. 11. Quiver plot comparison of CONS and EASY simulations on three workload traces.

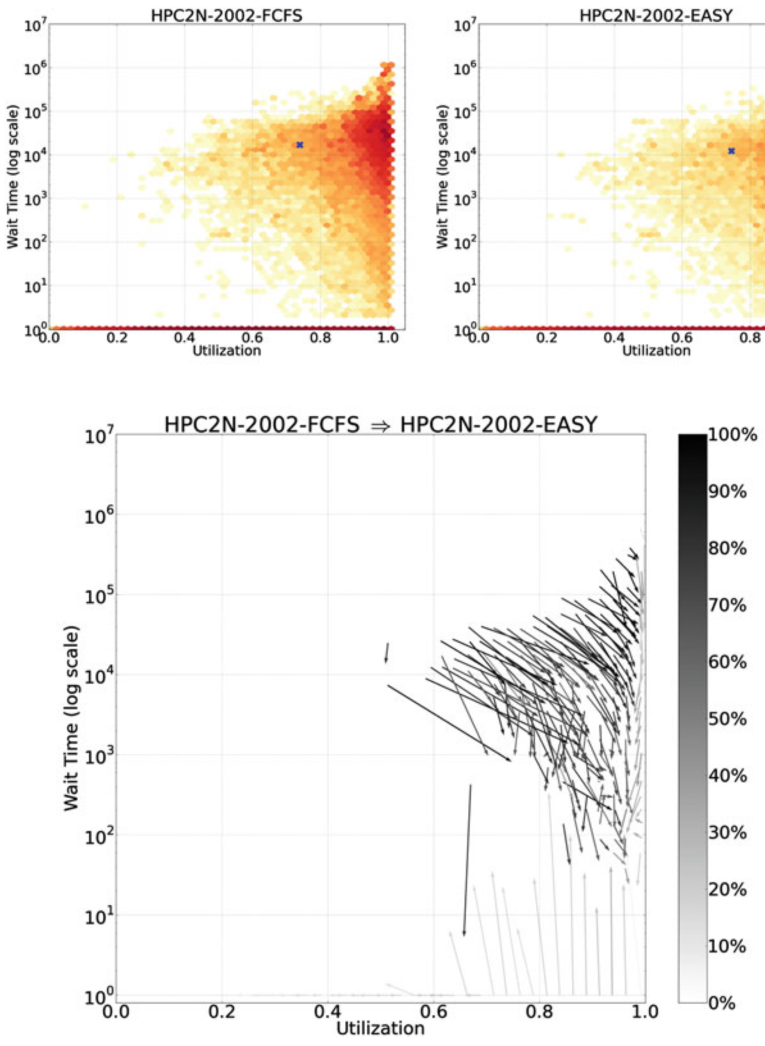


Fig. 12. Quiver plot comparison of HPC2N FCFS and EASY simulations.

a couple of cells with a downward arrow not following the general trend, representing a class of jobs that waited much less while experiencing on average the same load. Unlike backfilling comparisons, the edges of the plot are almost not affected: no "picket fence" is observed, even though there is a large number of jobs at the bottom, and jobs that experience 100 % utilization also do not move.

6 Conclusions

We suggested several different heatmap comparison techniques, some general image-based that are usable for any heatmaps (and in particular for performance heatmaps), and others that use specific domain knowledge on the underlying data and are only suitable for comparison of simulations based on the same underlying workload. *These techniques visualize how the whole distribution of performance results changes between the compared systems.* Thus they are significantly different from the common approach of comparing average performance values.

In some cases, looking at the whole distribution is crucial for reaching the correct conclusions. When conventional techniques are used, all the details are ultimately lost in favor of a single summary statistic. Thus a low Kullback-Leibler divergence (for example) is taken to mean that the compared distributions are similar. But this is not necessarily the case. It is also possible that some differences existed, but they canceled out. When looking at the whole distribution, these differences would be seen.

The image-based (**visual**) comparisons can be applied to any heatmaps, not only of parallel job scheduling evaluations. Moreover, they can even compare different job populations. In the context of job scheduling, such comparisons retain the ability of heatmaps to overcome the limitation of conventional average metrics, and provide a tool for comparing skewed distributions where average metrics are misleading.

Major advantages over the other methods are simplicity to compute and conciseness — a single heatmap is produced. Ratio difference visual comparison highlights more clearly acute differences between very similar job distributions, while simple difference is more representative of the overall changes. One disadvantage of using two colors is the inappropriateness for B&W printing or color-blind researchers. Gray levels may be used as a remedy.

A second group of methods is job aware plots. These can only be applied to the same job population. They are not applicable when the underlying data is different — e.g. to compare simulations based on a model that generates new jobs on each run. **Area plots** offer a high resolution view into different classes of jobs and can be used to dive in into edge cases. They do not allow for a single comparison representation, as each class is visualized separately. Thus area plots are complicated both to compute and to analyze.

Quiver plots employ averaging to plot trends of change. They create a single-image concise result, but as with any averaging, can be misleading when the average is not representative due to the presence of skewed distributions, as is the case with cells on the zero wait time axis. Quiver plots also suffer from noise in some of the logs checked. One possible remedy might be employing a low pass filter as a preliminary step.

For many cases the naive image difference comparison might be good enough, given its simplicity in computation and in visual understanding. For example, in the comparison of SDSC-SP2 EASY vs. EASY_P the visual comparison (Fig. 3) plainly shows that EASY_P reduces the number of high wait time jobs, which is exactly the basic result seen in the job-aware area plot and quiver comparisons (Figs. 7 and 10).

The job-aware comparisons allow for more advanced observations, at the possible expense of obscuring the general picture. In particular they can serve to understand some deeper behaviors:

- Looking into a job-aware comparison of a log's original scheduler versus a simulated one shows that the distribution of jobs remains similar in any large sub class of jobs. This shows that the simulation completely mixes the jobs and does not retain any relations with how they were treated in reality. This is a much stronger conclusion than can be derived based on the plainly visible difference between the heatmaps.
- Quiver plots show a "center of weight" that exists in backfilling based schedulers (both EASY and conservative backfilling). Jobs are drawn to it from all sides. This is a similar result to the postulation Rudolph and Smith made that schedulers have a "Desired Operating Range" (DOR) [12].
- Looking in both directions of comparison allows to highlight invariants in scheduler behavior. For EASY those are jobs with zero wait time or with 100 % experienced load. This might allow to distinguish more "interesting" jobs in the evaluation.

This paper is only a first try at heatmap comparison, which may serve as a basis for development of more advanced scheduler evaluation and comparison tools. Naturally, it would be interesting to apply the same techniques to a wider class of schedulers and environments. The techniques used can be extended, both in usability (how good is the visual representation) and applicability for different scenarios. One possible development of another type of comparison would be to compare time frames instead of comparing what happens to jobs: the change in a particular time window between two different scheduling runs. Another would be to group the jobs by different properties, such as the users who submitted them. As further research, heatmaps can also be drawn for different axes, for example wait time vs. job duration or job parallelism, allowing to observe correlations between performance effects and different job attributes.

Acknowledgements. Many thanks to all those who have made their workload data available through the Parallel Workloads Archive.

References

1. Anscombe, F.J.: Graphs in Statistical Analysis. Am. Stat. **27**(1), 17–21 (1973)
2. Feitelson, D.G.: Looking at data. In: 22nd International Parallel & Distributed Processing Symposium (IPDPS), April 2008
3. Chapin, S.J., Cirne, W., Feitelson, D.G., Jones, J.P., Leutenegger, S.T., Schwiegelshohn, U., Smith, W., Talby, D.: Benchmarks and standards for the evaluation of parallel job schedulers. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1999, IPPS-WS 1999, and SPDP-WS 1999. LNCS, vol. 1659, pp. 67–90. Springer, Heidelberg (1999)
4. Crovella, M.E.: Performance evaluation with heavy tailed distributions. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 2001. LNCS, vol. 2221, pp. 1–9. Springer, Heidelberg (2001)

5. Downey, A.B., Feitelson, D.G.: The elusive goal of workload characterization. Perform. Eval. Rev. **26**(4), 14–29 (1999)
6. Etsion, Y., Tsafrir, D., Feitelson, D.G.: Process prioritization using output production: scheduling for multimedia. ACM Trans. Multimed. Comput. Commun. Appl. **2**(4), 318–342 (2006)
7. Frachtenberg, E., Feitelson, D.G., Petrini, F., Fernandez, J.: Adaptive parallel job scheduling with flexible coscheduling. IEEE Trans. Parallel Distrib. Syst. **16**(11), 1066–1077 (2005)
8. Krakov, D., Feitelson, D.G.: High-resolution analysis of parallel job workloads. In: Cirne, W., Desai, N., Frachtenberg, E., Schwiegelshohn, U. (eds.) JSSPP 2012. LNCS, vol. 7698, pp. 178–195. Springer, Heidelberg (2013)
9. Lifka, D.: The ANL/IBM SP scheduling system. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1995. LNCS, vol. 949, pp. 295–303. Springer, Heidelberg (1995)
10. Feitelson, D.G., Rudoplh, L., Schwiegelshohn, U., Sevcik, K.C., Wong, P.: Theory and practice in parallel job scheduling. JSSPP 1997. LNCS, vol. 1291, pp. 1–34. Springer, Heidelberg (1997)
11. Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/
12. Rudolph, L., Smith, P.H.: Valuation of ultra-scale computing systems. In: Feitelson, D.G., Rudolph, L. (eds.) IPDPS-WS 2000 and JSSPP 2000. LNCS, vol. 1911, pp. 39–55. Springer, Heidelberg (2000)
13. Performance Heatmap Utilities. https://bitbucket.org/krakov/heatmaps