We thank the reviewers for their insightful and helpful comments. We repeat their comments here in purple; our own comments and responses are in blue.

Reviewer #1: I genuinely do not understand the contribution of this submission.

First of all, simplified versions of real-life applications where real calculations and data are replaced by time delays and dummy data correspondingly have been used for testing and debugging purposes since the advent of computer science. I have been using "skeleton" applications in my practice for decades. I have never found their writing/generation particularly challenging. The system presented in the paper provides some basic support in this direction. It is not clear however what is the advantage of using the proposed system instead of doing it yourself.

The authors claim as the main novelty the use of skeleton applications for performance analysis and optimization. I do not understand this part at all. The authors use the real-life application to build its skeleton on a given real-life platform. It is obvious that by fitting parameters you can make the skeleton run very similarly to the real application on this platform for the given data set. Even if you use a very simplistic performance model like the one in this work. So what? Why use the skeleton on this platform instead of the *available* real application at all? If the skeleton could be used on other platforms, where this application is not available, for its performance analysis, the approach would be justified. However, this type of use requires much more realistic performance modeling, taking into account the complexity of modern platforms where compute nodes typically integrate many heterogeneous and tightly coupled computing devices such as CPU cores and various accelerators. Correspondingly, this type of use is not addressed in the paper at all.

In summary, the submission presents a straightforward software tool, the theoretical and practical value of which are not clearly and convincingly articulated.

We agree that the theoretical basis of this work is not new; the idea of application replacements (models, kernels, benchmarks, etc.) has been used in computer science, particularly in benchmarking and designing new systems, for a long time, but usually in an ad hoc manner. However, the use of this concept in distributed computing is limited, particularly when applied in a systematic way. In addition, we don't know of another scalable solution that is available for distributed systems.

An example of the practical value of this work comes from a real use case with Berkeley. A company co-author Zhang is collaborating with has an application for which they want to choose which distributed file system and configuration will give best performance on a research benchmark system Berkeley is building. The company's policy does not allow them to share the application. But instead, they can share the I/O trace of that application. Berkeley is planning to build a skeleton application based on that trace and then to use the benchmark system to measure the performance of the file systems and configuration.

Reviewer #2: Major Comments:
The paper discusses the development of a tool for creating "application skeletons" as a surrogate for real applications for use in the system design process. The authors correctly identify the difficulties in using the real application for this purpose - access, complexity, legacy code etc. The need for such surrogates for use in system design and performance analysis is well reasoned. The paper presents a nice way of constructing such surrogates and ha interesting results. I would recommend the paper for publication after minor revision (see comments below).

This surrogate problem has been attempted from an algorithmic standpoint (see for e.g. "On the Characterization of OpenCL Dwarfs on Fixed and Reconfigurable Platforms" by Krommydas, Feng,

Owaida, Antonopoulos, and Bellas in the 25th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), June 2014.) where simpler versions of the algorithms at the core of popular applications are used. While that approach is theoretically more scalable it does not provide an easy process to use knowledge of actual runs of the application. The proposed blackbox approach here that attempts to deduce the "application skeleton" from performance data is an interesting alternative. I like the careful delineation of resource need and use and the phases of the computational tasks and their performance profile. Limitations of this approach are that if the scale of the problem to be solved changes significantly then the data driven approach is likely to
provide simple scaling which may not be true. The data driven modeling that this paper espouses is usually based on sound statistical principles. While, some statistical methodology is mentioned e.g. sampling, averages - clarification would be useful.

Three things that would improve the paper:
a) a careful discussion (short!) on the merits of the different approaches of constructing a "model".

We've created a new section after the introduction that includes discussion of different types of models (application replacements), and also includes the former Section 6 (related work).

b) Clear identification of the statistical processes used in model definition and testing

For each set of skeleton parameters generated manually, we ran 5 experiments and saw no significant variation, and we added this information to the paper. Both synapse and skeleton emulation are very consistent over multiple runs. We have added information on measured stdev Tables 2, 5 and 7. Our errors are dominantly systematic errors, and statistical errors are mostly negligible.

c) a clear justification for the choice of the 3 example applications

We've added a paragraph in Section 5 to explain and justify why we chose these applications.

Q. Are the extensions beyond the conference paper sufficient?
Yes. The automated parameter estimation is indeed a significant step beyond.

Minor Comments:
1. Typos in abstract - copy -paste issues likely.

This was indeed a copy-paste problem in the submission in the Elsevier system used for FGCS; it has been fixed.

2. Page 6 - awkward sentence please rewrite.
"We want to balance between mimicking the exact the operation sequence
of the real application tasks' I/O and computation in the skeleton task, which
requires reimplementing the actual task and results in poor programmability,
and having very simple programming but very different performance."

This has been fixed.

Reviewer #3: The authors propose a tool that called application skeleton to ease the creation of multistage workflow applications. The paper is easy to read, the choices made by the authors are justified, and the skeletons are validated against traditional applications, namely Blast, Montage and CyberShake.

C1. The proposal made by the author is a concrete one that is targeted at helping scientists deploy and configure distributed applications that usually require a lot of effort. I was however a bit surprised that the authors did not develop their solution as an extension of an existing popular workflow system, e.g., Taverna, Kepler or Triana. The description of Skeleton leaves the reader with the impression that it shares many of the properties of workflow systems. Using a popular workflow system would allow users to benefit from the functionalities provided by such systems, e.g., designing the distributed applications using a workbench. Besides, developing the solution as an extension of a popular workflow system means that the solution has chances of being adopted in practice by real users.

Skeleton is not a workflow system, it is a workload generator, and the workloads it generates can be run on a variety of systems, including workflow systems. We chose to support Swift and Pegasus, since we are most familiar with them, but we believe the idea is very general, and since the Skeleton code is fully open source, someone else could choose to add support for Taverna, Kepler, or Triana.

C2. One of the difficulties underlying staging is identifying the values of the staging parameters. The authors present two approaches manual and automatic, using the SYNAPSE tool. However, the authors do not provide enough details for the reader to grasp how SYNAPSE identifies the values of the parameters.

We expanded the description of SYNAPSE's data collection by describing the operational mode of the profiler. It is difficult to provide more detail of the process without discussing the implementation of the tools, which we feel is out of scope for this particular paper.

C3. The authors need to specify precisely in the introduction what was published previously and what has been added in this paper. They are presented together in the current version with no distinction.

We have added more detail in the introduction, just before the contribution bullets.

C4. The related work section is thin. The authors simply say that the work is novel and therefore there is no much to compare with. I am not convinced. Deploying and configuring distributed and parallel applications has been an issue for a long time. And I believe that there are approaches that although do not tackle the same problem as the authors are related. For example, Holl et al. made a proposal where the results of identifying the parameter values that yield an optimal execution of a scientific workflow are shared.

Sonja Holl, Daniel Garijo, Khalid Belhajjame, Olav Zimmermann, Renato De Giovanni, Matthias Obst, Carole A. Goble: On specifying and sharing scientific workflow optimization results using research objects. WORKS@SC 2013: 28-37

In general, we accept that researchers have been using application replacements for experiments for a long time, and some of those replacements were certainly tunable in one way or the other. The distinction in our work is that the Skeletons provide that capability in a systematic way, being tunable to a range of diverse application, including multi-stage applications, with the promise to preserve the significant part of the application's behavior. This is why there are not many references: the problem space is pervasive, but attempts to solve it systematically are rare, and even more so for distributed applications. We have added this statement to the related work section, and because we are interested in what work there is, we have added the suggested citation as well.

Minor comments:
===============

C5. Page 3: We previous presented -> previously

This has been fixed.

C6. Page 3, end of paragraph 2: what are dd commands

This has been fixed.