Best Practices for Using Proxy Applications as Benchmarks
(the slides are available under “Presentation Materials” in the above URL)
Date: April 15, 2020
Presented by: David Richards (LLNL) and Joe Glenski (HPE)

Q. For proxy apps that implement some exact/polynomial time algorithm, discerning FOM may not be as difficult as problems that implement some heuristics. Like a number of graph applications could be based on heuristics, which makes formulating FOM challenging IMO (Traversed-Edges-Per-Second popularized by Graph500 may not be relevant for all graph problems). Do you have a comment about how to deal with such problems where there is essentially no consensus on FOM formulation?

A. You’re right that defining an FOM is harder for some problems than for others. Probably the best you can do when there is no consensus is to work with Subject Matter Experts (SMEs) and identify and propose something that makes sense. If nothing else, (problem size)/(wall time) can be a place to start.

Q. Sorry I missed the first few minutes of your talk. I gather that you are trying to compare the performance of a benchmarking tool which you call proxy with the real system. Proxy is a model. In order to compare, you have to know how the real system performs. If you already know how the real system performs, why do you need a model?

A. Sometimes you **do** want to use the real application to measure performance. However, proxies are essential in cases where the real application can’t be shared such as proprietary, export controlled, or classified codes. Proxies can also be useful when the benchmark process involves changing the code. Proxies tend to be much easier to modify than production application codes.

Comparing the proxies to the real application is an important step in assessing the fidelity of the proxy as a model for the real application.

Q. These tables are symmetric. Could you compare visually 2 systems by having the data from one in the upper triangular section and the other in the lower one?

A. That’s an interesting idea that we hadn’t thought of. We’ll have to try it to see how well it works.

Q. The selective metrics may be platform specific. Do you regenerate the set for each platform?

A. Yes.

Q. What is the granularity of the similarity? whole proxy application vs. whole parent application. Or kernels vs. kernels?

A. The data presented here is for the full application. We hope to do kernel-to-kernel comparisons in the future.

Q. Can you comment on how we can validate that the numerical results of a proxy application are equivalent to the parent application?

A. Proxies aren't necessarily intended to produce equivalent numerical results to their parents. For example Quicksilver has so many simplifications in the Physics of the problem compared to Mercury that the two codes can't really even run the same problems. Usually, we're trying to demonstrate that proxies and parents stress hardware in similar ways. We think the cosine similarity metric is one way of demonstrating that. Of course it is still important to verify that a proxy operates correctly according to its specifications. Some proxies, such as LULESH solve problems that also have an analytic solution making such verification easy. In other cases, programmers need to be more creative in establishing verification procedures.

Q.Could you give an example of the Law of Unexpected Consequences on slide 33?

A. Sure, An organization released an RFP for a large HPC system. The RFP had a fixed budget and delivery date (which is common) and also a requirement to demonstrate at least a certain level of High Performance Linpack (HPL) result. Unfortunately, the budget did not provide enough funds for a balanced system that could provide that level of HPL performance. In order to achieve the required HPL result, the vendors had to propose configurations that had very low interconnect network capability, limited storage bandwidth and space, and minimum software tools. While the resulting system did run HPL at the required level, the system did not perform well for many actual applications. While the RFP authors had not set out to create an unbalanced system, that was the result.

C. huge +1 to using real apps as procurement benchmarks from a vendor who is more than happy to do projections on 1MLOC Fortran legacy codes.

A. As mentioned above, using real apps in a benchmark can be the right choice. We turn to proxies when production apps can't be used.

Q. If we discover a similarity issue between a proxy and an app, could we tell why**?**

A. The cosine similarity metrics we have described could probably help provide clues. Other performance tools could also help. However, I would expect that any real explanation would always come down to an SME looking at the codes to isolate the root cause of any differences.