# HPC Production Job Quality Assessment

Omar Aaziz, Jonathan Cook (advisor)
Department of Computer Science, New Mexico State University

All About Discovery!™
New Mexico State University

## Motivation and Goals

- Help large numbers of novice-to-average HPC users to learn about their application run by aiding them with immediate feedback.
- Point HPC administrators to possible system anomalies.
- Evaluate the performance of a particular run whether it is healthy, abnormal, or critical.
- Assess job performance with near-zero overhead technique, suitable for production environment.
- Achieve low runtime estimation error.

## Analysis Model

1. Divide the entire application runtime into 8 equal-length intervals.
2. For each interval *i*, compute the average for each hardware counter metric *M*, for each process *p*.

$$M_{i,p} = avg(intervalSamples)$$

3. Combine the individual process interval averages by computing their mean and standard deviation.

$$M_i = avg(M_{i,p}) \ over \ p$$

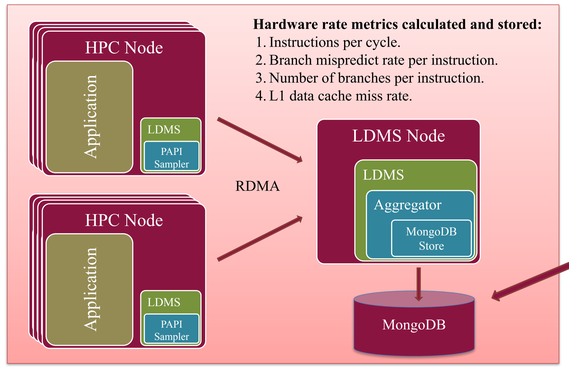
$$SD_i = stddev(M_{i,p}) \ over \ p$$

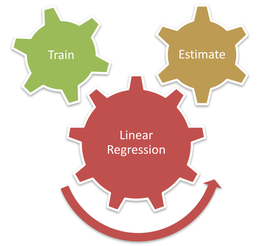
4. With four hardware counters rate metrics being used, eight intervals, and two values per interval per metric (mean and std-dev), we have 64 values, calculate the Principal Component Analysis (PCA) and select the N top components.
5. Apply linear regression over the set of runs being analyzed.

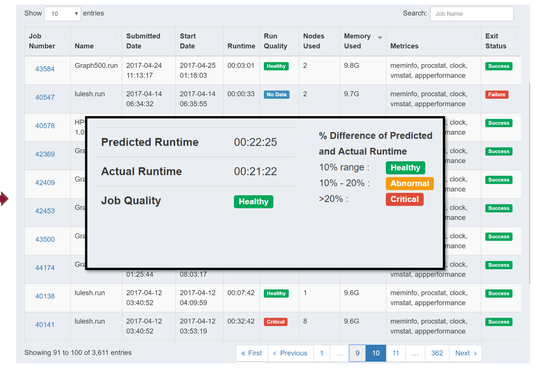
$$ExpectedRuntime = LReg(\frac{ProbSize}{\#Proc}, \frac{ProbSize}{\#Nodes}, \frac{\#Proc}{\#Nodes}, ProbSize, PC_1, ..., PC_n).$$

6. On new runs, capture data and calculate metrics, then us PCA and regression formulas to estimate expected runtime.
7. Compare expected runtime to actual runtime and assign job quality category.

## Methodology



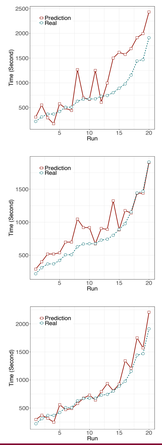



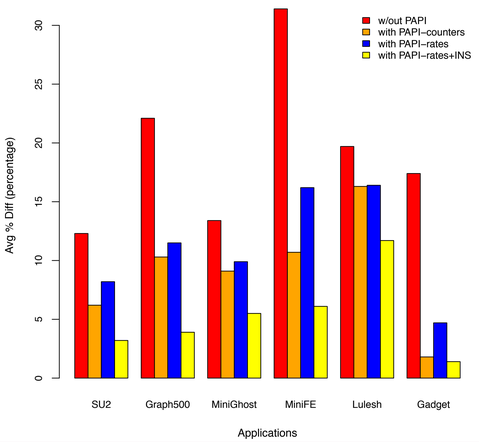

## Experiment Results

**Testbed Specification**

10-node cluster, each node has two Intel 12 core CPUS (e5-2695) and 256GB RAM, and are connected with an InfiniBand interconnect.

| Applications Name | Runs | | Parameter Ranges | | | PCA Number |
|---|---|---|---|---|---|---|
| | Train | Test | Processors | Node | Problem Size | |
| MiniFE | 206 | 20 | 16-128 | 1-8 | 134M-719M | 7 |
| SU2 | 98 | 20 | 27-216 | 1-9 | 48-84 | 5 |
| Gadget | 167 | 20 | 16-128 | 1-8 | 120K-202K | 7 |
| MiniGhost | 202 | 20 | 8-125 | 1-8 | 125M-512M | 6 |
| Graph500 | 191 | 20 | 16-256 | 1-8 | 24-29 | 6 |
| Lulesh | 216 | 20 | 8-216 | 1-9 | 13K-46K | 7 |




## Conclusion

- Estimated the expected runtime of a particular run of an application through basic job parameters and rate metrics derived from raw PAPI hardware counters.
- Used a standard statistical model rather than a potentially more powerful but opaque machine-learning method.
- Analysis was integrated into online job statistics interface that can be queried as the job runs, just as other job statistics are.

## Future Work

- Characterize application similarities, and whether proxy applications are similar to the applications they claim to represent.
- Compare how a deep learning technique will perform compared to our statistical model.

## Acknowledgments

- We gratefully acknowledge the support of James Brandt, John Noe, and Ann Gentile from Sandia National Laboratories.
- This work was also supported by the NSF under grant CNS-1337884