Methods for Automatic Behavioral Stratification in Scientific Codes

Todd Gamblin
tgamblin@cs.unc.edu

Rob Fowler
rjf@renci.org

Daniel Reed
Dan_Reed@unc.edu
**Problem:** Low-overhead performance monitoring and characterization of large-scale scientific applications

**Approach:**
1. Use clustering algorithms to discover behavioral equivalence classes of nodes in scientific applications.
2. Discovered classes provide insight into application behavior across nodes.
3. Variance of monitored data within classes is also low, and we exploit this property to reduce monitoring overhead.
Adaptive Sampling

• For large clusters, cannot collect performance data from all nodes
  • Centralized collection is impossible at scale!

• We’ve developed the Adaptive Monitoring and Profiling Library (AMPL)
  – Models parallel application as a population of runtime performance events
  – User specifies desired confidence and error in advance

• Uses population sampling to estimate performance metrics
  – We give a probabilistic guarantee that confidence and error will be met
Adaptive Sampling

Given:
- Desired confidence
- Desired minimum error

Based on variance of collected data:
- Determine the minimum number of events to sample

Collect data from just enough nodes that minimum number of events are collected

Update sample size as monitoring proceeds, depending on variance
- Updates are done at the end of fixed time windows

\[ m = \frac{Ns^2}{N^2V^2 + s_r^2} \]
\[ V = \left( \frac{d}{z_{\alpha}} \right)^2 \]

For \( N \) performance events, error \( d \), and confidence interval \( z_{\alpha} \)
Results with sPPM

- 5-14% of data overhead with PAPI hardware counter measurement with sPPM at 90% confidence and 8% accuracy
- 7-14% of overhead at 99% confidence and 1% error for low-variance metrics
Application Signatures

- RENCI has developed an Application Signature library
- Signatures are a compact representation of application trace data
- Trace data is collected from runtime instrumentation, fit according to custom filters
  - Exhaustive data is not stored
  - Only curve fit, generated dynamically
Application Signatures

- Per-node application signatures can be generated via tools like SvPablo and TAU
  - Allow performance and behavior of nodes of the same application to be compared
- Signature library is written in C++, can be linked with most HPC applications
  - Supports custom data compression filters, fast least-squares is provided
  - Available from RENCI CVS
Signatures as behavioral model

- Many scientific codes have behavioral equivalence classes of nodes
  - May be caused by
    - load imbalance
    - functional decomposition
    - performance problems
- Automatically finding these equivalence classes will help us understand
  - how an application is behaving
  - how well it is performing
Clustering

- Clustering algorithms such as k-medoids and CLARA can be used to automatically find these groups from performance data from scientific applications.

- We take signatures collected from scientific codes and cluster them to find equivalence classes for entire application runs:
  - At scale, users need these methods to understand performance of codes.

- Individual node-level understanding is not enough.
Clustering

Per-node, per-metric Application Signatures

Clustering (k-medoids, CLARA)

Different colored regions can be monitored separately
Guided Stratification

• We know from above that number of nodes sampled depends on variance of monitored data

• Adaptive sampling can be improved by stratifying population
  • Total nodes sampled will be smaller if variance within monitored groups is less than variance between them:

\[
\frac{1}{M} \sum_{i=1}^{k} (M - M_i)S_i^2 < \sum_{i=1}^{k} M_i(\bar{Y}_i - \bar{Y})^2
\]

• Intuitively, sample size is lower when sum of intra-group variances is smaller than variance of population as a whole
Guided Stratification

• Procedure:
  – Cluster per-node application signatures
  – Use output of clustering to separate nodes into behavioral equivalence classes
  – Re-run sampling with this new stratification

• 2 benefits:
  – Adaptively sampling groups separately can further reduce overhead in number of nodes sampled
  – Provides a longitudinal behavioral picture of an application’s performance that isn’t possible with other tools

Each discovered equivalence class is monitored separately.
Total number of monitored nodes is reduced further.
Subgroups have lower variance than group as a whole, so total number of sampled nodes is decreased.
Monitoring PAPI Metrics in sPPM: Monitored Sample Sizes for 1-16 clusters

- Clustering done on raw data here (exhaustive trace)
- 99% confidence, 1% error
Monitoring PAPI Metrics in sPPM: Monitored Sample Sizes for 1-16 clusters

- Clustering done on application signatures here (compressed trace)
- 99% confidence, 1% error
Work in Progress

• Extensions of Existing Work:
  – Precise measurement of overhead and perturbation of applications
  – Tests at larger scale (>1024 nodes)
  – Tests with more applications (ADCIRC, Chombo)

• Additions:
  – Clustering on sampled data rather than exhaustive data
  – Dynamic repartitioning of nodes in response to runtime phase changes
  – Learning methods for selecting most relevant metrics to partition on
  – Finding outliers in existing groups and relocating them as behavior changes