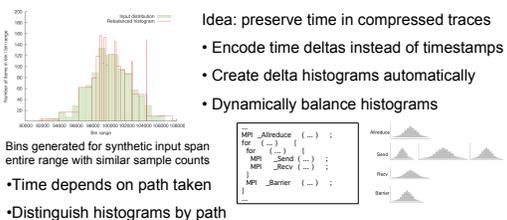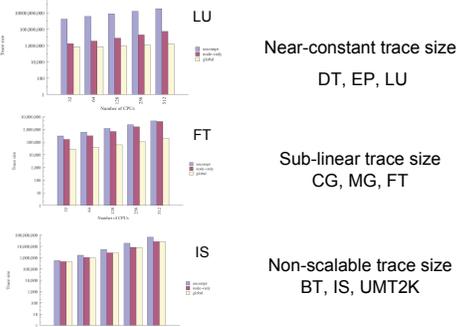# An Open Framework for Scalable, Reconfigurable Performance Analysis

Todd Gamblin[‡], Prasun Ratn[*†], Bronis R. de Supinski[*], Martin Schulz[*], Frank Mueller[†], Robert J. Fowler[‡], Daniel Reed[‡]
[*]Lawrence Livermore National Laboratory, [†]North Carolina State University, [‡]Renaissance Computing Institute
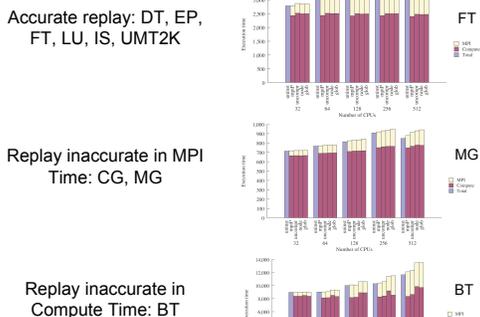
ScalaReplay: Replay Using Histogram Timing Annotations

Idea: preserve time in compressed traces
- Encode time deltas instead of timestamps
- Create delta histograms automatically
- Dynamically balance histograms

Bins generated for synthetic input span entire range with similar sample counts

- Time depends on path taken
- Distinguish histograms by path

Number of histograms per record depends on number of possible call paths

Histograms detect imbalances

Variable sizes capture variance

Sample bimodal distribution from UMT2K collectives

Trace sizes (NAS Benchmarks and UMT2K)

Near-constant trace size
DT, EP, LU

Sub-linear trace size
CG, MG, FT

Non-scalable trace size
BT, IS, UMT2K

Replay accuracy (NAS Benchmarks and UMT2K)

Accurate replay: DT, EP, FT, LU, IS, UMT2K

Replay inaccurate in MPI Time: CG, MG

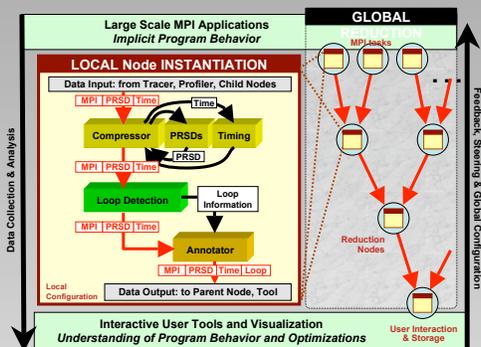Replay inaccurate in Compute Time: BT

Description of problem

- Size of machines is rapidly increasing (130,000+ processors)
- Tools will be overwhelmed with data
- Need scalable, online measurement and analysis

ScalaTrace: Reconfigurable, Scalable Performance Analysis

Trace representation using Power Regular Section Descriptors (PRSDs)

- Near-constant low-overhead MPI traces
- Annotate with additional reconfigurable data, *e.g.*
  - Time using adaptive histograms (left)
  - Progress rates, load imbalance (right)

Future Directions

- **Flexible framework for application-specific tools**
  - Ability to select compression schemes for different fields, data types
  - Foster combination, interaction between data collection, analysis mechanisms
  - Adaptive, self-tuning runtime systems

- **Near term:**
  - Adaptive wavelet transform topology, equivalence class detection
  - Full time-sequence compression

## Evolutionary Load-Balance Analysis with Scalable Data Collection

Idea: Normalize measurements and models based on application semantics.

Progress loops
- Typically outer loops in SPMD codes
- Indicate absolute progress towards some domain-specific goal
- Basis for comparison of load over time

Effort Loops
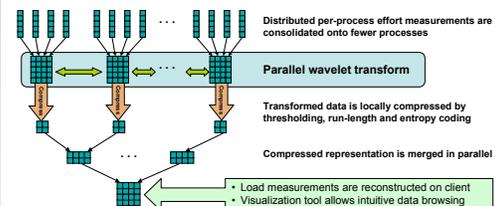- Variable-time loops, represent load
- Data-dependent execution
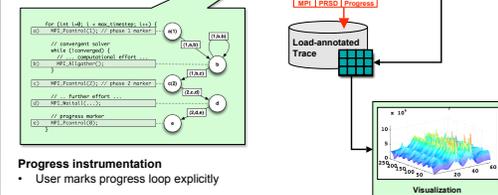
Progress instrumentation
- User marks progress loop explicitly

Effort modeled with code regions
- Dynamically detected at runtime
- Split MPI-op trace at collectives and wait operations
- User can further divide code into phases with instrumentation

Scalable Compression Using Wavelet Transform

Distributed per-process effort measurements are consolidated onto fewer processes

Parallel wavelet transform

Transformed data is locally compressed by thresholding, run-length and entropy coding

Compressed representation is merged in parallel

- Load measurements are reconstructed on client
- Visualization tool allows intuitive data browsing

Load Balance in ParaDiS

Models dislocation dynamics in crystals
- Dislocations discretized as *nodes* & *arms*
- Recursive spatial domain decomposition
- Balancer subdivides nodes/arms along x, then y, then z