Scalable Compression and Replay of Communication Traces in Massively Parallel Environments

Michael Noeth†, Frank Mueller†, Martin Schulz*, Bronis R. de Supinski*

†Department of Computer Science, North Carolina State University, Raleigh, NC

*Lawrence Livermore National Laboratory, CASC, Livermore, CA

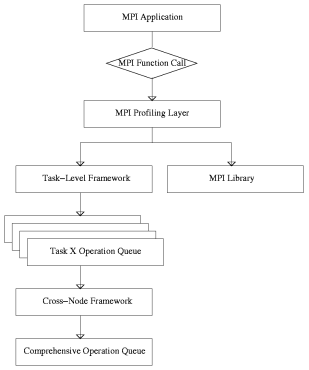
Problem Motivation

- How can communication traces be gathered in peta-scale computing?
  - need scalable, loss-less approach → objective: ***near constant-size traces***
  - help understanding communication patterns → not easy!
  - assist in procurement → rapid prototyping of communication needs
- Current communication analysis tools fall in 2 classes:
  1. aggregation methods → lossy (e.g., mpiP)
  2. flat traces → loss-less but not scalable (e.g., Vampir)

Our Approach

Record Traces
- Use MPI profiling layer
- Compress at task level
- Compress at node level

Replay Traces
- Inverse of merging algorithm

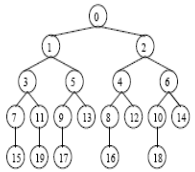


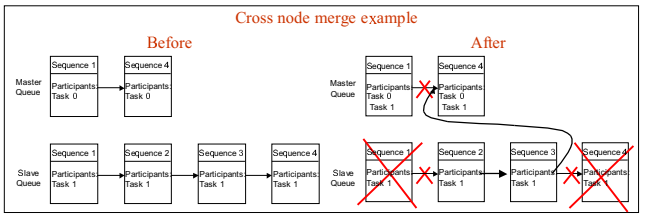
Task level compression framework
- *Umpire:* PMPI wrapper generator
  - Initialization wrapper
  - Tracing wrapper
  - Termination wrapper
- Task-level compression of MPI calls
  - Provides load scalability
- Interoperable w/ cross-node framework

Cross-Node Framework Interoperability
- Single Program, Multiple Data (SPMD) nature of MPI codes
  - Maintain structure of calling sequences
    - stack walk signatures
- Match operations across tasks by manipulating parameters
  - Source / destination offsets
  - Request handles
- Event aggregation
  - Special handling of MPI_Waitsome

Cross-Node Compression Framework
- Invoked after application termination
- Merges operation queues produced by task-level framework
- Job size scalability

Reduction over binary radix tree
- Cross-node framework merges operation queues of each task
- Merge algorithm supports merging two queues at a time
- Radix layout facilitates compression (constant stride b/w nodes)
- Need a control mechanism to order merging process





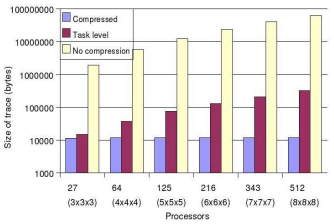
Temporal cross node reordering
- Requirement: queue maintains order of operation
  - Unmatched sequences in slave queue always moved to master
  - Results in lower compression rate
- Solution: only move operations that must be moved
  - Intersect task participation lists of matched & unmatched operations
  - Intersection empty → no dependency
  - Otherwise →ops must be removed

Replay Mechanism
- Motivation: Possible to replay traces on any architecture
  - Useful for rapid prototyping→procurement
  - Communication tuning (Miranda -- SC'05)
  - Communication analysis (patterns)
- Replay Design
  - Replays comprehensive trace produced by recording framework
  - Parses trace, loads task-level op queues (inverse of merge algorithm)
  - Replay on-the-fly (inverse of compression algorithm)

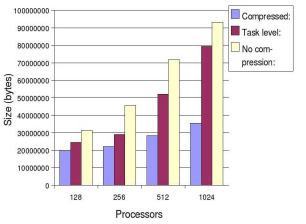
Experimental Results

Near constant size for fully compressed traces

Trace File size – 3D Stencil (Log scale)



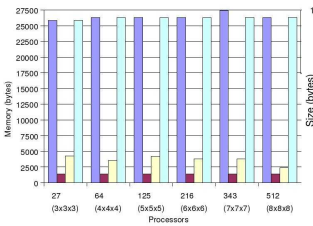
Trace File size – Raptor (Linear scale)



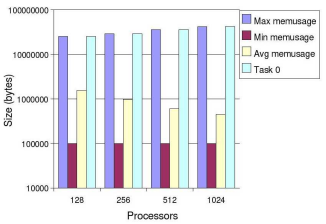
Near constant memory consumption (per node) for fully compressed traces
- max ~ task 0, min = leaves, avg = middle layer (decr. w/ node #)

Average memory consumption decreases w/ more processors

Memory Consumption – 3D Stencil



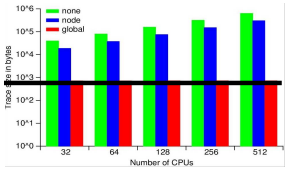
Memory Consumption – Raptor



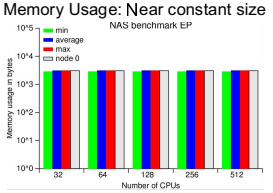
NAS PB experiments, codes fall into 3 classes:

1. Constant size traces: EP, IS and DT

Trace file size: Near constant size

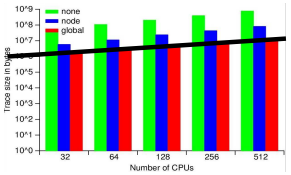


Memory Usage: Near constant size

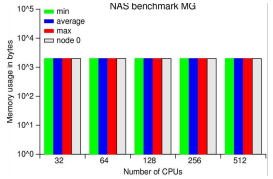


2. Sub-linear traces: MG, LU

Trace file size: sub-linear

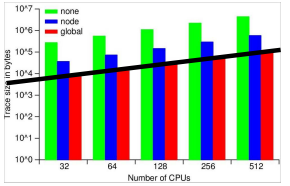


Memory Usage: Sub-linear

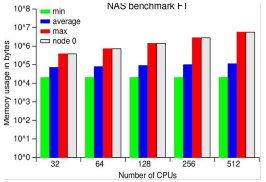


3. Non-scalable traces: FT, BT, CG

Trace file size: not (yet) scalable
- Still 2-4 orders of magn. smaller
- Due to complex comm. patterns along diagonal of 2D layout
- Could be improved



Memory Usage: Non-Scalable

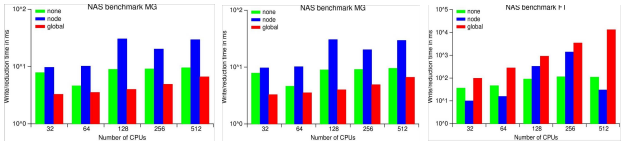


NAS PB Codes – Output times

EP, IS, DT – near constant
Sometimes fast cross node compr

MG, LU sub-linear
Still fastest cross-node compr.

FT, BT, CG – Non-scalable
Non-competitive (better write node compressed)



Contributions and Future Work
- **Scalable approach to capture full trace of communication**
- **Scalable replay mechanism**
- **Trace analysis → determine inefficient MPI usage**
- **Assist in procurement via rapid replay**
- **Use to address task mapping problem**

† Supported in part by NSF CVS-0310203, CCF-0429653, CAREER CCR-0237570

* Performed under auspicies of US DOE by UC California LLNL contract # W-7405-Eng-48, UCRL-POST-225759