Extrapolation of and Code Generation from Communication Traces

Frank Mueller
North Carolina State University

NC STATE UNIVERSITY
Department of Computer Science

NORTH CAROLINA STATE UNIVERSITY
FOUNDED MARCH 7, 1887

Agenda

- Overview of ScalaTrace
- Probabilistic Tracing and Replay (ICPP'11)
- Communication Extrapolation of Traces (PPoPP'11)
- Generation of Executable Specifications from Traces (ICS'11)
- Automatic Benchmark Code Generation from Traces

Introduction

- Contemporary HPC Systems
  - Size > 1000 processors
  - take IBM Blue Gene: ~74k nodes, ~300k cores
- Challenges on HPC Systems (large-scale scientific applications)
  - Communication scaling (MPI)
  - Communication analysis
  - Task mapping
- Procurements require performance prediction

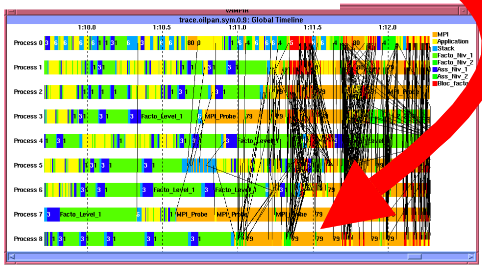
Communication Analysis

Existing approaches and short comings

- Source code analysis
  - + Does not require machine time
  - - abstr. level: apps complex, no dyn. info
- Lightweight statistical analysis (mpiP)
  - + Low instrumentation cost
  - - only aggregate information
- Slicing
  - + Fast
  - - lacks temp. info
- Performance Modeling
  - + Abstract, somewhat general
  - - only high-level stats, arch.-dependent
- Fully captured (lossless): Vampir, VNG
  - + Full trace available for offline analysis
  - - not scalable (n traces) / need viz cluster

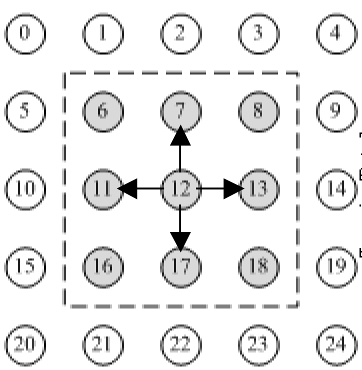
viz I/O

nodes

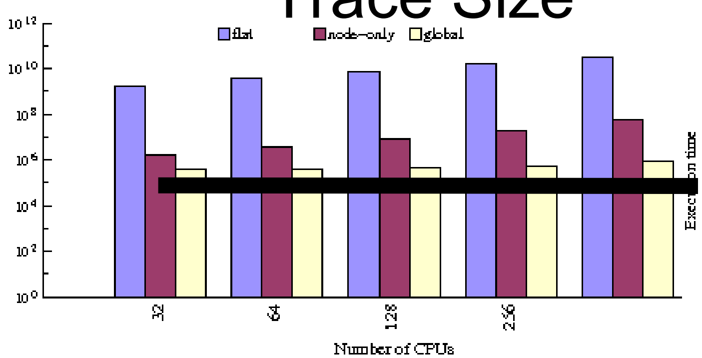


ScalaTrace: Lossless & Scalable Tracing

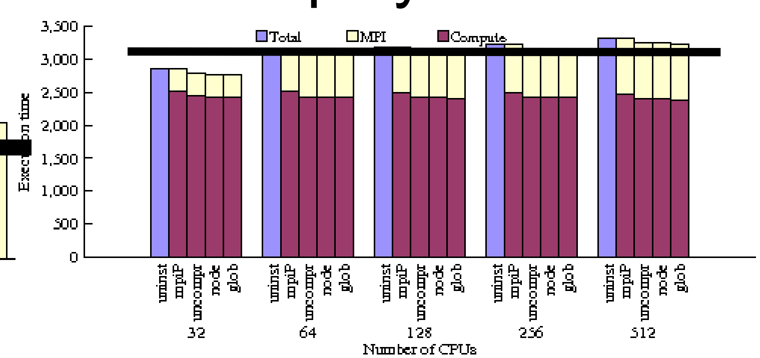
- Traces communication + I/O of MPI codes via interpositioning
- Trace size: Near constant size, one file represents all nodes
  - Intra-node (loop) & inter-node (task ID) compression of SPMD codes → preserves program structure
  - Location-independent encoding → scales
    - E.g., <10, MPI_Irecv(LEFT), MPI_Isend(RIGHT)>
  - Communication group encoding
    - <dim start_rank iteration_length stride {iteration_length stride}>
- Preserves timing for computation & communication (histograms)



Trace Size

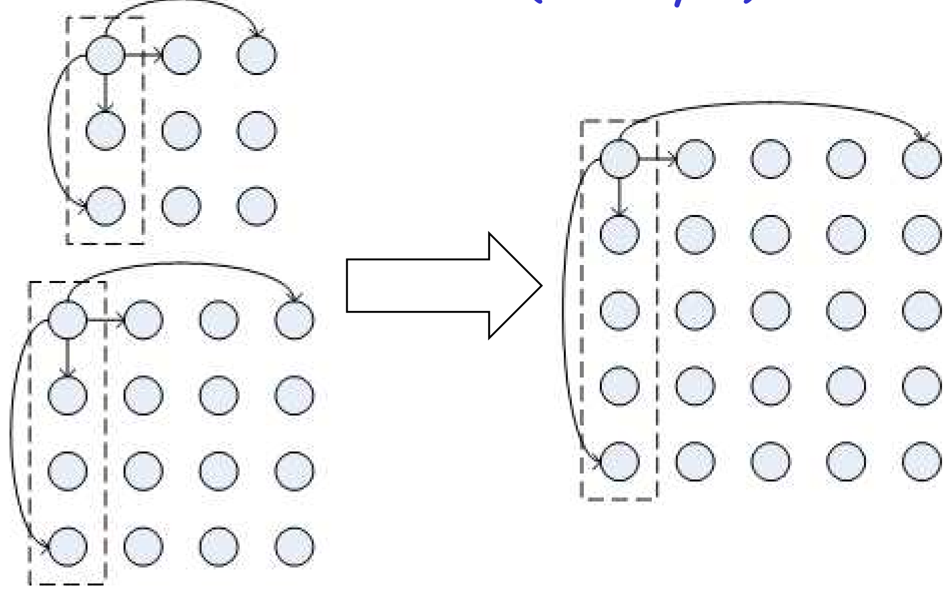


Replay Times



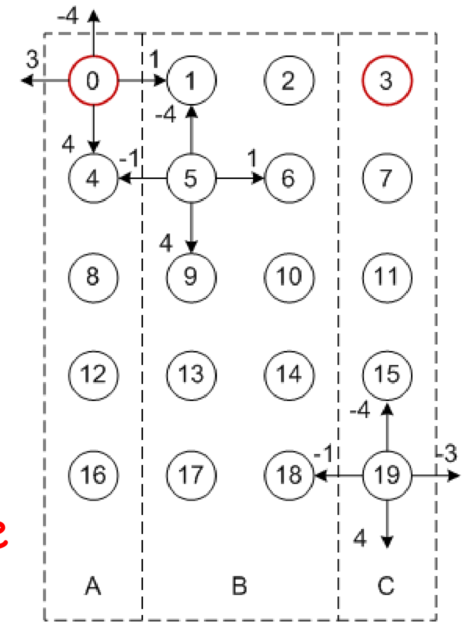
(1) Communication Extrapolation

- Motivation
  - Communication analysis at scale – without running app!
  - Modeling for procurements
  - Extrapolation on a single workstation!
- Idea: synthetically generate communication traces:
  - k small traces from app → large traces
  - E.g., P=8,16,32,64 nodes trace → P=4096 trace (or any P)
- Replay large trace/analyze it
- Challenges:
  - Topology detection
  - Message payloads
  - Time extrapolation



Identify & Extrapolate Comm. Topology

- Constrained to row-major stencil/mesh codes
- Algorithm for topology identification:
  - Partition nodes
    → groups according to comm. endpoints
  - Identify critical nodes
    @ corner/boundary of topo space
  - Calculate boundaries sizes of topological space
    - @ dimension i: Si = ni / ni-1 (need i+1traces)
- Communication trace extrapolation:
  - Extrapolate comm. params (SRC, DEST, CNT, …)
  - Extrapolate comm. group

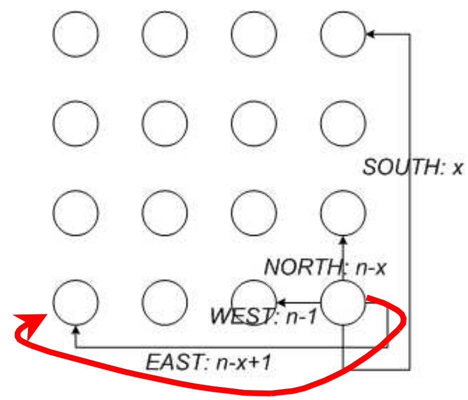


Outline of Algorithm

- Input: topology data (x, y, z) from detection phase
- Represent comm. param./group data as f(x,y,z) + unknown coeff.
  - $A*(xyz) + B*(xy) + C*x + D = V$
- Correlate multiple traces & construct set of linear equations
  - $A*(x_1 y_1 z_1) + B*(x_1 y_1) + C*x_1 + D = V_1$
  - $A*(x_2 y_2 z_2) + B*(x_2 y_2) + C*x_2 + D = V_2$
  - $A*(x_3 y_3 z_3) + B*(x_3 y_3) + C*x_3 + D = V_3$
  - $A*(x_4 y_4 z_4) + B*(x_4 y_4) + C*x_4 + D = V_4$
  - Recall: $n_3=xyz$, $n_2=xy$, $n_1=x$
- Gaussian Elimination to solve set of equations (solve A B C D)
- With known coefficients, substitute x,y,z at target problem size
- Calculate the desired param. value/comm. group data (V)

Example: Comm. Param. Extrapolation

- Assuming trace inputs for P = 16, 25, 36, 49
- Problem: extrapolate relative position of EAST for bottom-right node in a 10x10 mesh



N1=16: $x_1=y_1=4,\ z_1=1,\ v_1=13$ N2=25: $x_2=y_2=5,\ z_2=1,\ v_2=21$

N3=36: $x_3=y_3=6,\ z_3=1,\ v_3=31$ N4=49: $x_4=y_4=7,\ z_4=1,\ v_4=43$

N_t=100: $x_t=y_t=10,\ z_t=1,\ \mathbf{v_t=?}$

⇩

$Ax16 + Bx4x4 + Cx4 + D = 13$

$Ax25 + Bx5x5 + Cx5 + D = 21$

$Ax36 + Bx6x6 + Cx6 + D = 31$

$Ax49 + Bx7x7 + Cx7 + D = 43$

⇨

$A+B = 1$

$C = -1$

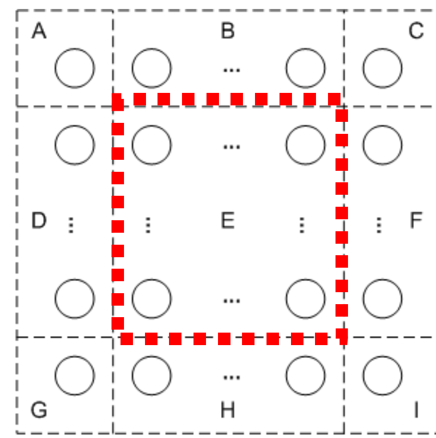
$D = 1$

⇨

$\mathbf{v_t = Ax100 + Bx10x10 + Cx10 + D = 91}$

Example: Comm. Group Extrapolation

- Ranklist of nodes:
  <dim,start_rank,iteration_length,stride,{iteration_length,stride}>
- Extrapolation performed for
  - start_rank
  - iteration_length
  - stride
- Consider ranklist of Group E



N_1=16: <2 5 2 4 2 1>	N_2=25: <2 6 3 5 3 1>
N_3=36: <2 7 4 6 4 1>	N_4=49: <2 8 5 7 5 1>

⇩

<2 x+1 x-2 x x-2 1>

⇩

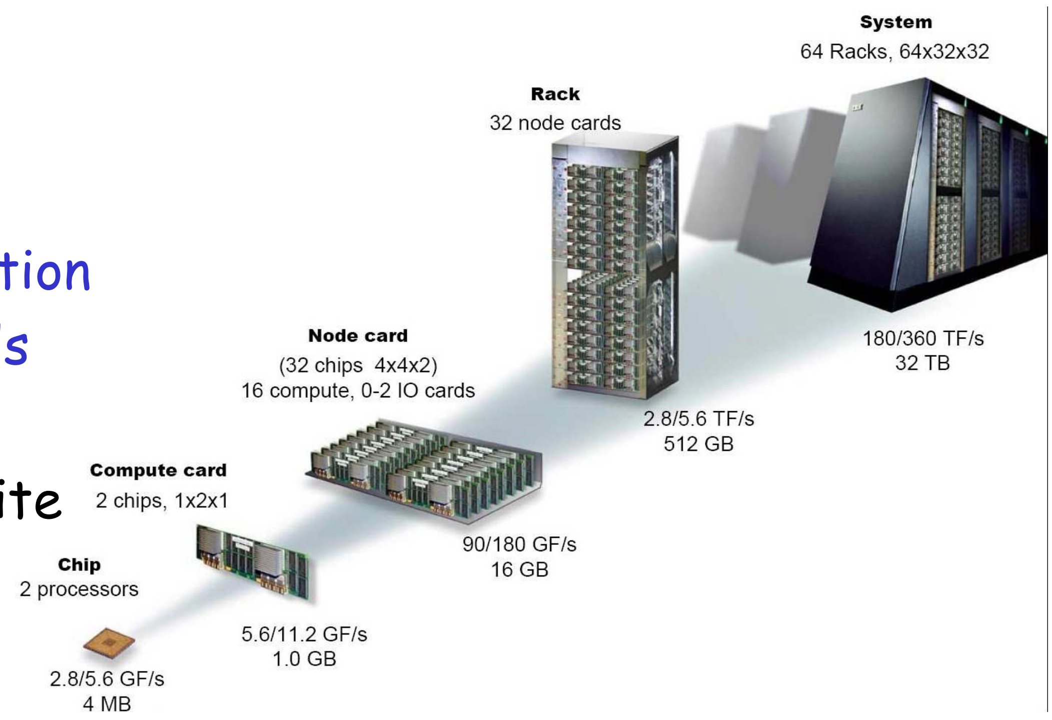
N_t=100: <2 11 9 10 8 1>

Timing Information Extrapolation

- Match delta times for same computation phase
  - in traces of different size (# nodes)
- Curve fitting to capture variations of delta times
- Currently: 4 types of timing trends
  - Constant: $t=f(n)=c$ → based on std. dev.
  - Linear: $t=f(n)=an+b$ → use least-squares method to fit curve
  - Inverse Proportional: $t=f(n)=k/n$ → std. dev. of $t*n$ (const. k)
  - Inverse Proportional + Constant: $t=f(n)=k/n+c$
    → least-squares for $t'=tn=cn+k$
- Utilize loosely defined criterion to select best fitting curve
  - Metric: std. dev./avg.

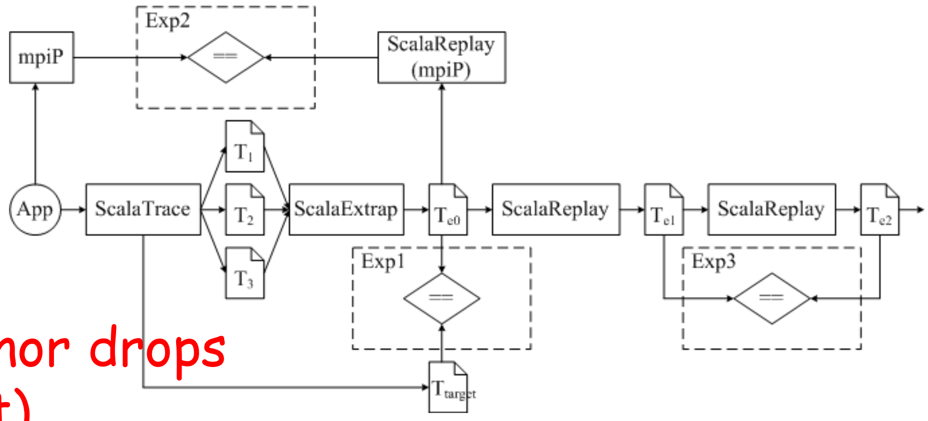
Experiment Framework

- Platform: Jugene (IBM BG/P)
  - 73,728 compute nodes and 294,912 cores
  - 2GB memory per node
  - 3D torus and global tree network
- Use subset of nodes
  - base trace generation
  - results verification
- Extrapolation algorithm
  - runs on a single workstation
  - requires only 1-2 seconds
- Experiments with NAS Parallel Benchmark (NPB) suite
  - version 3.3 for MPI



Correctness of Comm. Trace Extrapolation

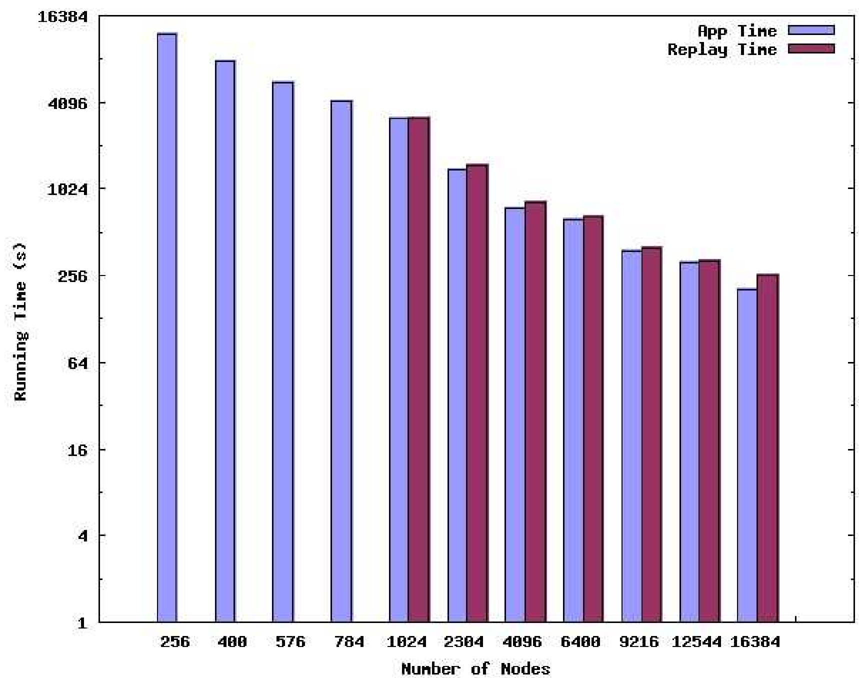
- for NPB: BT, EP, FT, CG, LU, IS
- Verified in multiple ways:
  1. Extrapolated trace = app trace @ same # nodes size & inputs
  2. Replay extrapolated trace w/ mpiP
     — Aggregate stats = stats from app execution
  3. Replay extrapolated trace w/ ScalaTrace → new trace
     — replay new trace, repeat until fixed point found



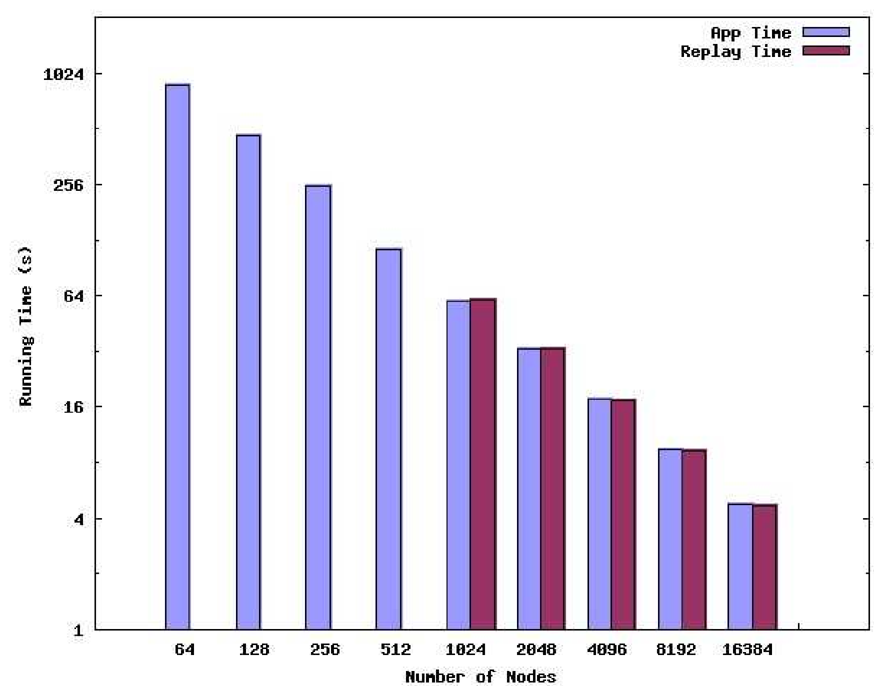
- Results:
  ✓ extrapolated = app traces
  ✓ Aggregate stats match
  ✓ ScalaReplay neither adds nor drops any comm. events (fixpoint)

Accuracy of Timing Extrapolation (i)

- for NPB: BT, EP, FT, IS, CG
- up to 16k nodes (not cores)
- t(extrapolated replay) = t(app)
- Accuracy = |Replay Time - App Time|/App Time) → generally>90%



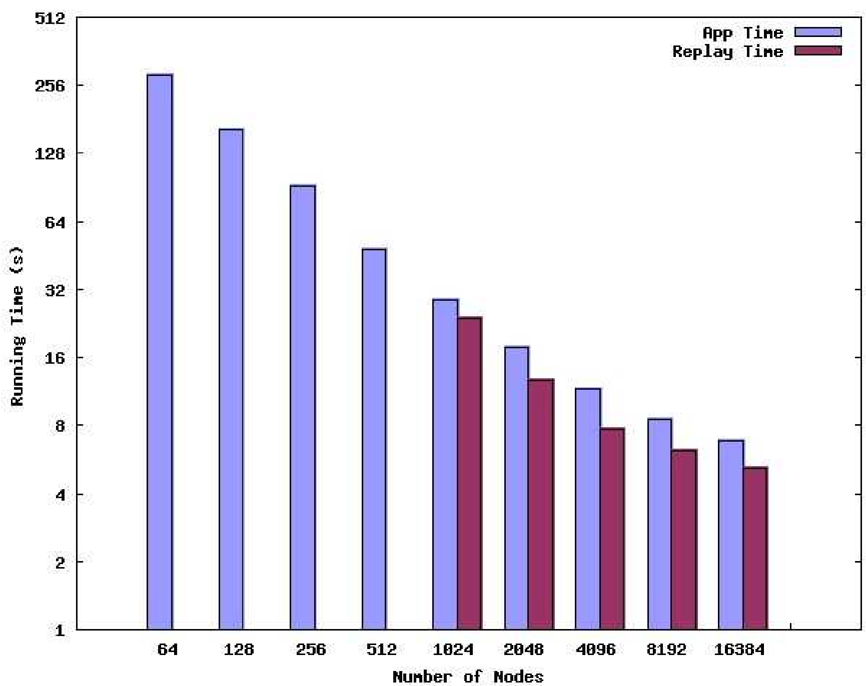
BT (class E)



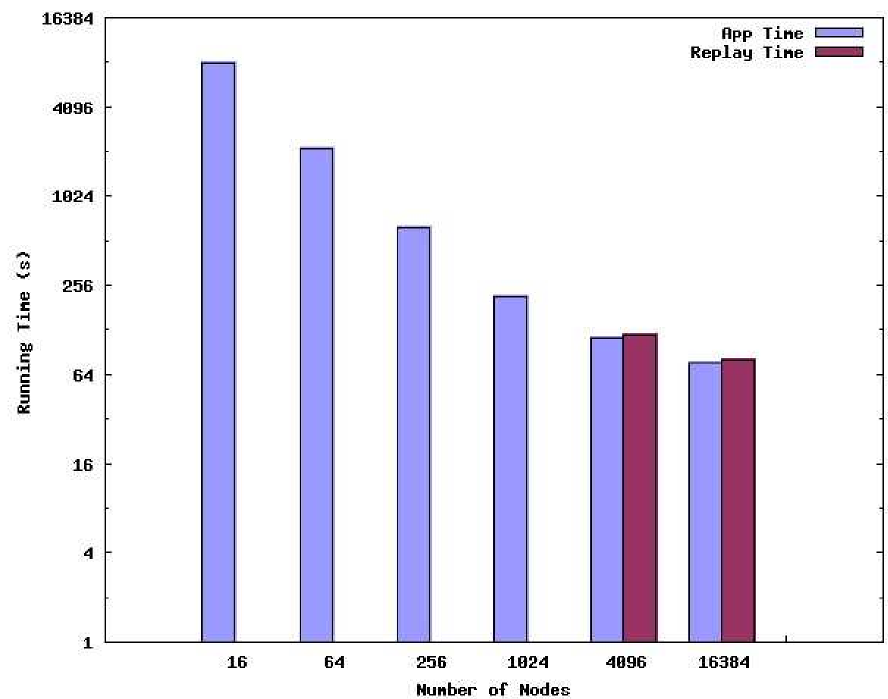
FT (class D)

Accuracy of Timing Extrapolation (ii)

- for NPB: BT, EP, FT, IS, CG
- up to 16k nodes (not cores)
- t(extrapolated replay) = t(app)
- Accuracy = |Replay Time - App Time|/App Time) → mostly > 90%



IS (modified input)



CG (class D) square topo

Conclusion for Extrapolation

Contributions:

- Algorithms & techniques for comm. extrapolation, handles
  - trace events
  - execution times
- Based on app runs at smaller scale
- Extrapolation shown to be
  - correct
  - accurate
- Obtains comm. behavior of parallel app at arbitrary scale
  - without actual execution at this scale → unprecedented
- Future Work:
  - Weak scaling

Extrapolation not so elusive anymore

(2) Communication Benchmark Generation

- Goal: Generate comm. benchmarks from apps that are
  - easy to 1) distribute, 2) use, 3) modify
- Extracted benchmarks from applications are
  - Performance-accurate
  - Application logic is stripped out
  - Readable, portable, modifiable
    - Collectives are consolidated
    - Nondeterminism has been eliminated
- Target coNCePTuaL: language for rapid generation of network benchmarks
- Compiler + runtime library

```
for(i = 0; i < 10; i++){
   MPI_Irecv(10, LEFT);
   MPI_Isend(10, RIGHT);
   MPI_Waitall();
}
```

⇒

```
For 10 repetitions {
   All tasks t asynchronously send a
      10-byte message to task t+1 then
   all tasks await completion
}
```

Code Generation from Application Trace

Application → Application Trace → Benchmark in coNCePTuaL

```
MPI_Init();
func1();
for(i = 0; i < 10; i++){
    func2();
    MPI_Irecv(LEFT);
    func3();
    MPI_Isend(RIGHT);
    func4();
    MPI_Waitall();
}
func5();
MPI_Barrier();
MPI_Finalize();
```

```
RSD1:   <All, Init>
        T1
PRSD1:  <10, RSD2, RSD3, RSD4>
        T2
RSD2:   <All, Irecv, LEFT>
        T3
RSD3:   <All, Isend, RIGHT>
        T4
RSD4:   <All, Waitall>
        T5
RSD5:   <All, Barrier>
RSD6:   <All, Finalize>
```

```
All tasks compute for T1 seconds
For 10 repetitions{
    All tasks compute for T2 seconds
    All tasks t asynchronously receive an x-byte
        message from task t-1 then
    All tasks compute for T3 seconds
    All tasks t asynchronously send an x-byte
        message to unsuspecting task t+1 then
    All tasks compute for T4 seconds
    All tasks await completion
}
All tasks compute for T5 seconds
All tasks synchronize
```

- All comm. events & computation times generated
- Benchmark contains loop structure
  → easier to read/modify than translation to straight-line code

Consolidating Collectives

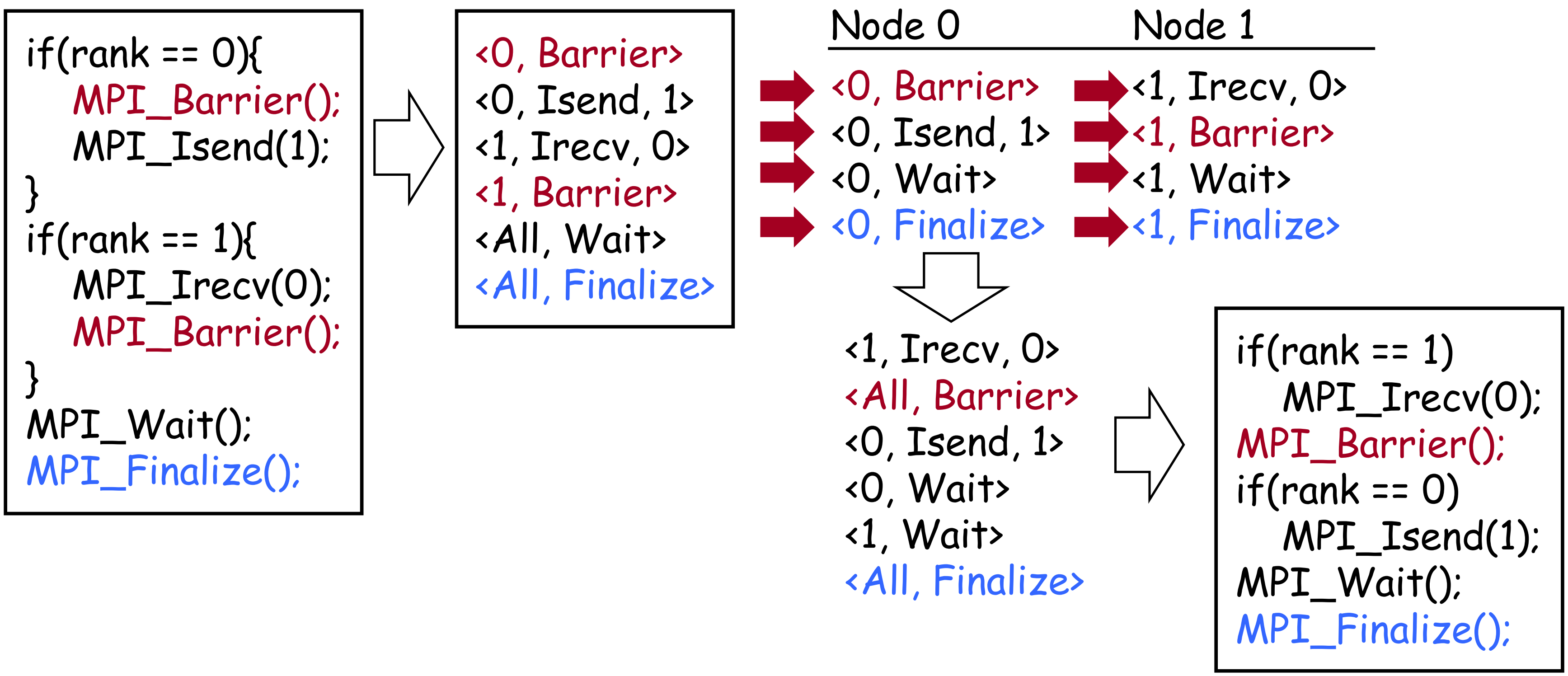
- MPI collectives have context sensitive semantic
  - Multiple statements ←→ single collective
  - Harms readability!

```
if(rank == 1)
   MPI_Barrier();
if(rank == 0){
   MPI_Barrier();
   MPI_Isend(1);
}
if(rank == 1){
   MPI_Irecv(0);
   MPI_Barrier();
}
if(rank == 0)
   MPI_Barrier();
MPI_Wait();
```

```
if(rank == 1)
   MPI_Irecv(0);
MPI_Barrier();
if(rank == 0)
   MPI_Isend(1);
MPI_Wait();
```

- Traverse trace with a single node
- Align collectives → generate new trace from old one; retain compression
  - Generate RSDs for non-collective events
  - Block & context switch execution at collectives
  - Generate RSD for collective only when last node arrives
  - On-the-fly loop compression

Example: Consolidating Collectives



- On-the-fly loop compression → ensures scalability of trace size

Eliminating Nondeterminism

- Wildcard receives
  → nondeterministic execution

```
MPI_Recv(..., MPI_ANY_SOURCE, ..., status)
if( status.MPI_SOURCE == 0 )
   <Do some LONG-running computation>
else
   <Do some SHORT-running computation>
```

- Bad performance reproducibility!
- Bad readability and modifiability!
- Replace MPI_ANY_SOURCE with arbitrary valid sender

Traverse trace with a single node

Match point-to-point comm.

- Traverse, context sw. if blocked @
  - Blocking Send/Recv
  - Wait, Waitall, etc.
  - Collectives, Finalize
- Switch to node that unblocks current one
- Pair events & unblock an "MPI process" if possible
- Replace MPI_ANY_SOURCE w/ rank of 1st matching sender

Eliminating Nondeterminism (Example)

- No decompression. No new trace generated.

Node 0	Node 1	Node 2
Send(1)	Irecv(2)	Send(1)
Finalize()	Isend(2)	Recv(1)
	Waitall()	Finalize()
	Recv(0) / Recv(ANY_SOURCE)	
	Finalize()	

Pending Matches	Pending Matches	Pending Matches
	0, Send	1, Irecv 1, Isend

Eliminating Nondeterminism (Example)

- Deadlock detection: no additional event could be unblocked during last traversal round (across events of all MPI tasks)
- Example: broken program that may deadlock

Node 0	Node 1	Node 2
Send(1)	Recv(0) / Recv(ANY_SOURCE)	Send(1)
Finalize()	Recv(0)	Finalize()
	Finalize()	

Cycle Identified!!!

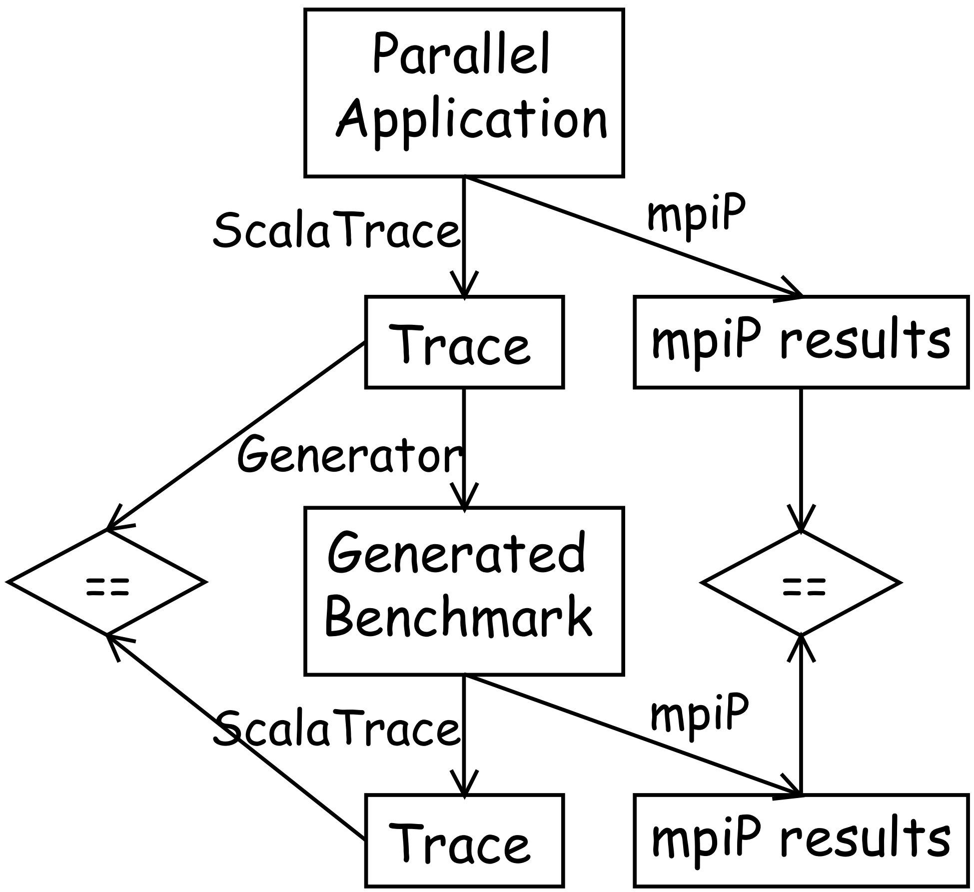
- Above method represents a sufficient condition → depends upon the order of traversal
  - If cycle found → deadlock; o/w unknown

Experimental Environment

- Platform:
  - Ocracoke: IBM Blue Gene/L
    2,048 nodes, 1GB memory/node
  - ARC: 108 nodes, 1728 cores, 32GB mem/node
  - Single workstation for benchmark generation
- Benchmarks:
  - NAS Parallel Benchmark (NPB V3.3 MPI)
    - Class C inputs, Strong Scaling
  - Sweep3D, Weak Scaling

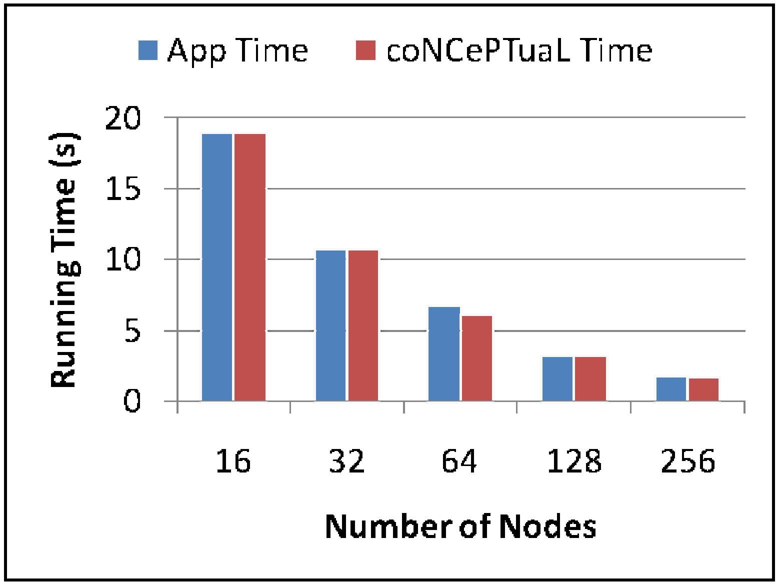
Communication Correctness

- Equivalent traces
  - Communication pattern is preserved
- Equivalent mpiP result
  - Same number of events
  - Same data volume sent
- LU: wildcard receives
- Sweep3D: per-node collective

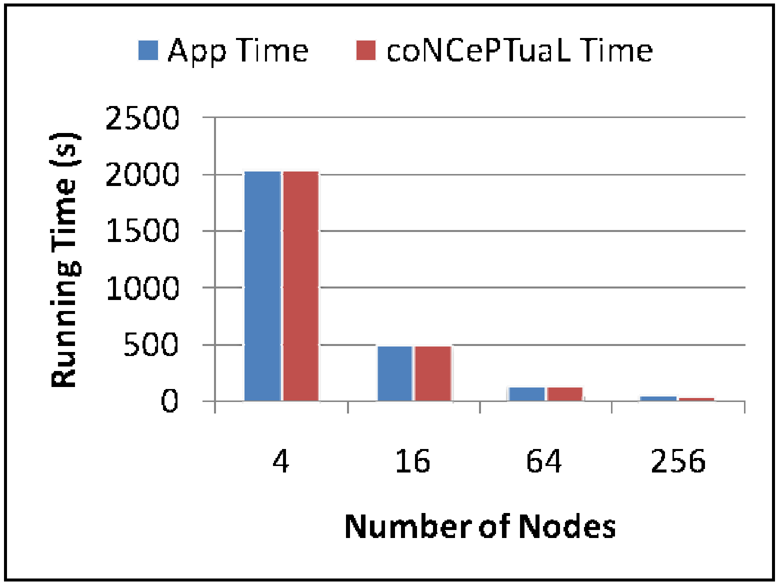


Accuracy of Generated Timings

- Time the application and the generated benchmark, and compare the results
- Mean absolute percentage error is only 2.9% → formula: |TcoNCePTuaL – Tapp|/Tapp x 100



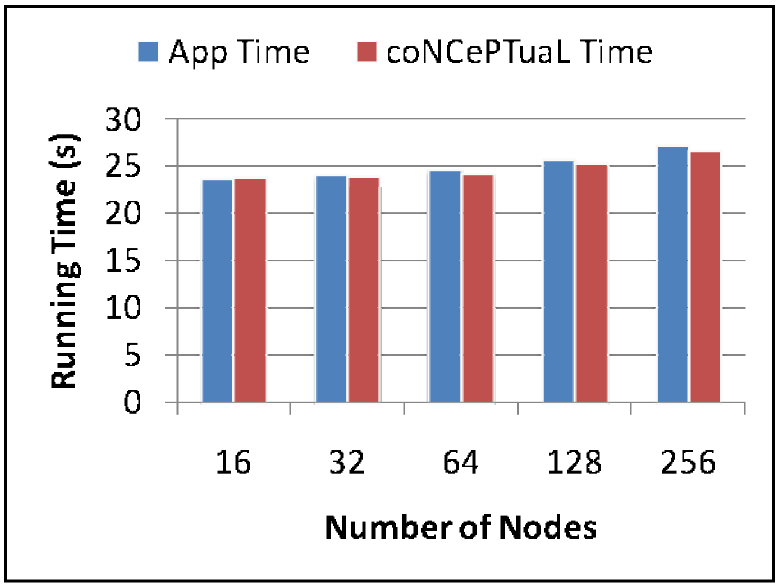
IS



LU

Accuracy of Generated Timings

- Time the application and the generated benchmark, and compare the results
- Mean absolute percentage error is only 2.9% → formula: |TcoNCePTuaL – Tapp|/Tapp x 100

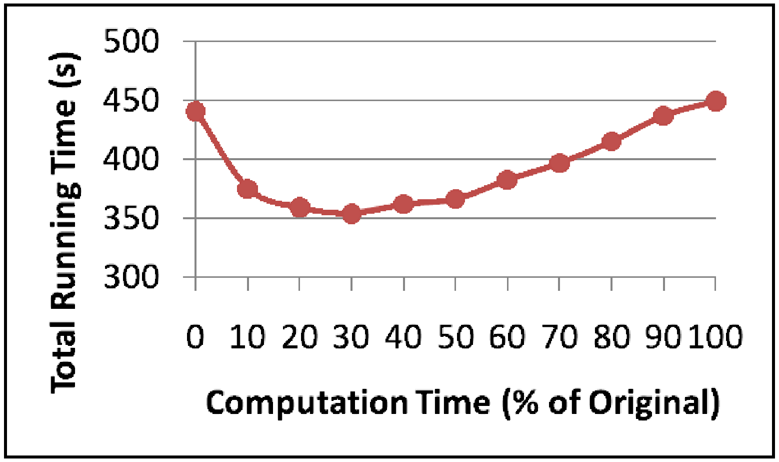


Sweep3D: Weak Scaling

Applications of the Benchmark Generator

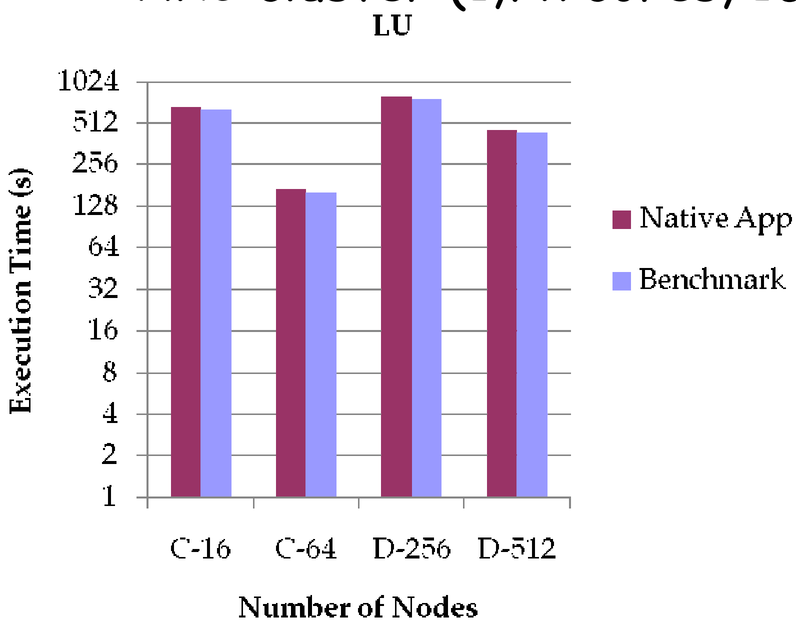
- Determine limits of computation/comm. overlap or effect of computational acceleration (e.g., GPUs)
  - Experiment: ARC cluster, 64 cores, Ethernet, BT b'mark
  - Shorten the spin times gradually
    - 100%: original compute overhead (simulated w/ spin)
    - 0%: no compute overhead → infinitely fast processor

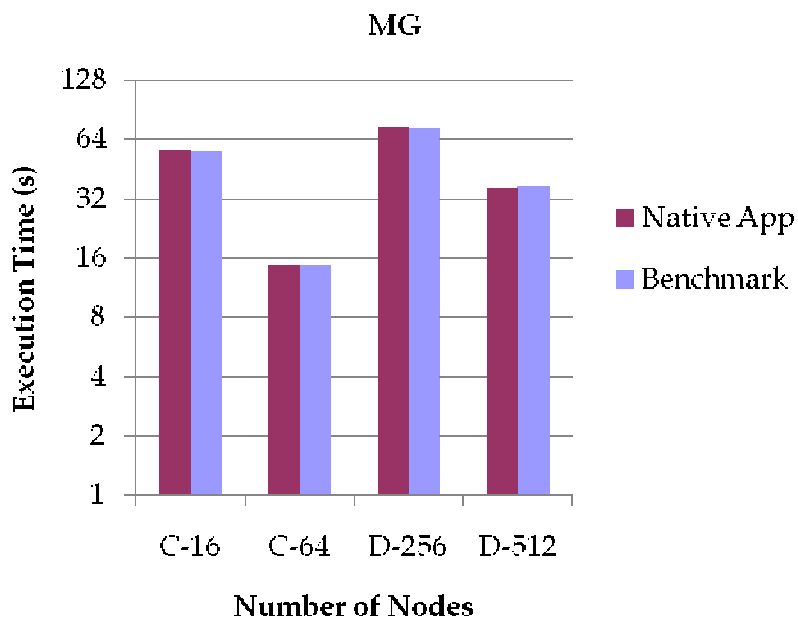
- Best speedup: ~3X → overall runtime reduced by 22%
- 0%: network contention or extra memory copies
- Platform-specific result



(3) C Benchmark Generation Results – Timing Accuracy

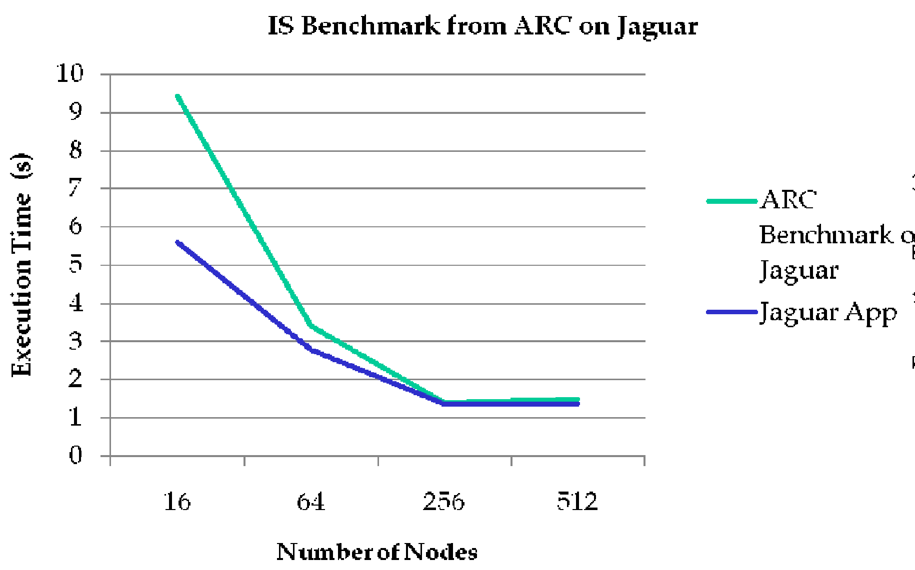
- time from Init → Finalize for app and benchmark
  |Tgen – Tapp|/Tapp x 100
- Timing accuracy = ~ 6.7% (avg. error)
- ARC cluster (1,7k cores, 16 cores/node Opteron, 32 GB RAM)

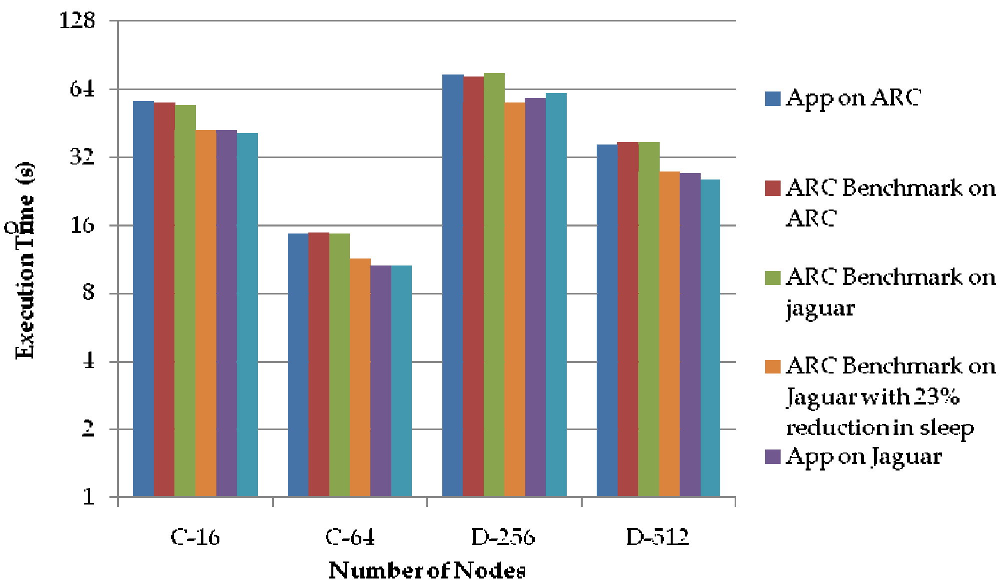




Cross Platform Results: ARC vs. Jaguar

- Jaguar ~23% faster execution on compute kernels
- Resemblance to benchmark obtained on ARC
  - IS: strong scaling reduced per node work
    → close match @ 256 tasks
  - Nearly perfect match after 23% speed correction





Summary

- Trace extrapolation feasible → exascale modeling
- An automatic communication benchmark generation framework
  - ScalaTrace → coNCePTual or C
- Generate benchmarks from real-world apps
  - Ensure performance fidelity, abstract away application logic
  - Readable, portable, modifiable, reproducible
    - Consolidation of collectives
    - Elimination of nondeterminism
  - Obfuscates code → for restricted source code access
  - Facilitate "what-if" analysis

Acknowledgements

NC STATE UNIVERSITY
Department of Computer Science

- Xing Wu (NCSU, intern @ LANL)
- Scott Pakin (LANL)
- Mike Noeth, Prasun Ratn (NCSU, interns @ LLNL)
- Martin Schulz, Bronis R. de Supinski (LLNL)
- Karthik Virjayakumar (NCSU, intern @ ORNL)
- Phil Roth (ORNL)

- best paper [IPDPS'07,JPDC], timed replay [ICS'08], I/O [PDSW'09], extrap [PPoPP'11], gen. specs [ICS'11], prob. Replay [ICPP'11]
- Code availabile under BSD license:

  moss.csc.ncsu.edu/~mueller/ScalaTrace

- Funded in part by Humboldt Foundation, NSF 0937908, 0429653, 0410203, CAREER 0237570, 0958311
- Part of work … auspices … U.S. DoE by UC-LLNL under contract No. W-7405-Eng-48 + UT-Batelle, LLC DE-AC05-00OR22725 + Sandia DE-AC52-06NA25396.