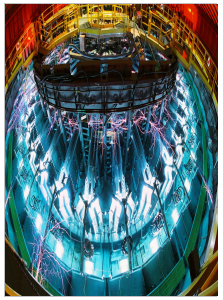


It's not the heat, it's the humidity

Scheduling resilience activity at scale

Patrick M. Widener
Kurt B. Ferreira
Scott L. Levy

Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA
patrick.widener@sandia.gov

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

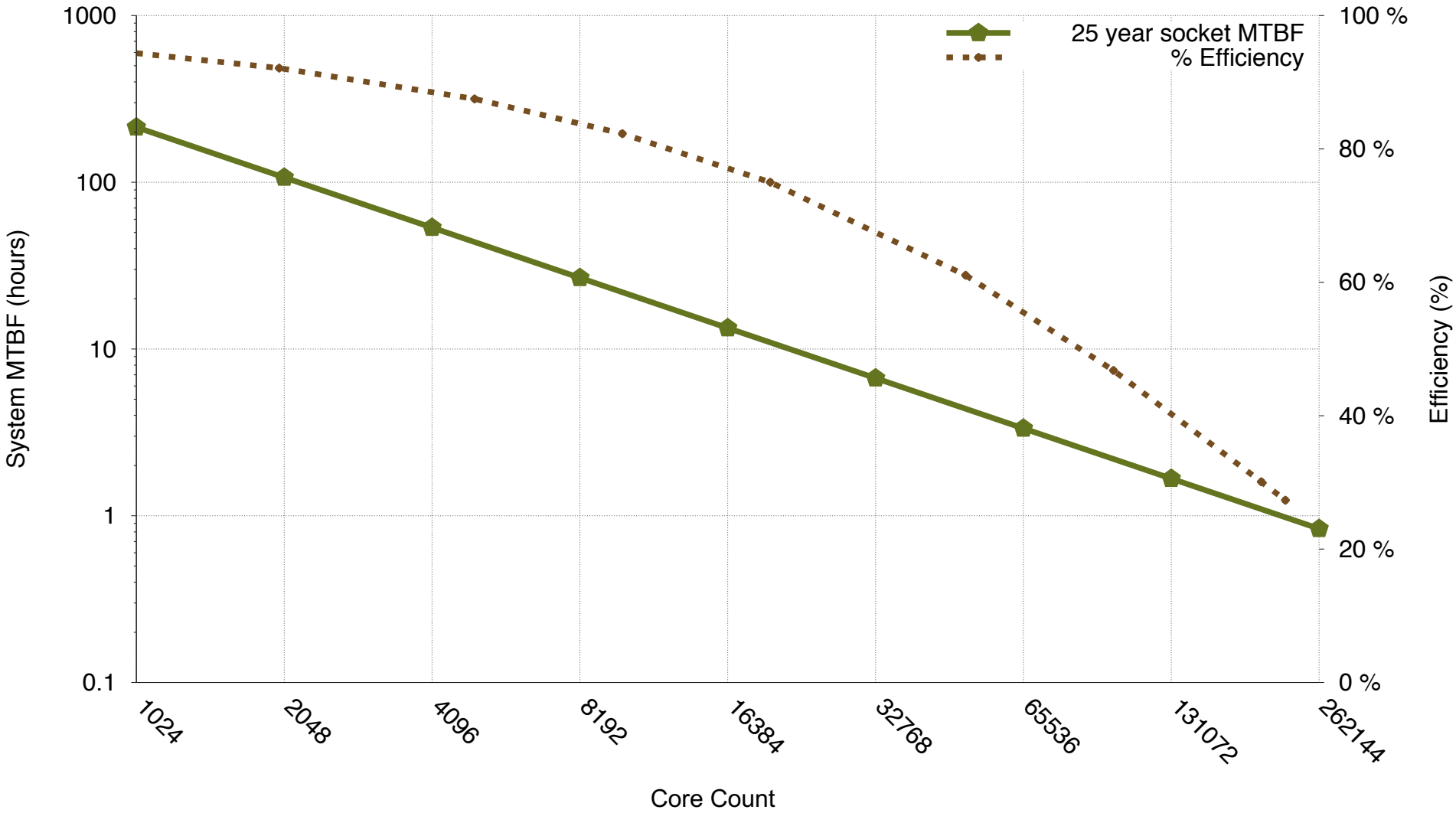
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.
SAND2017-5143C

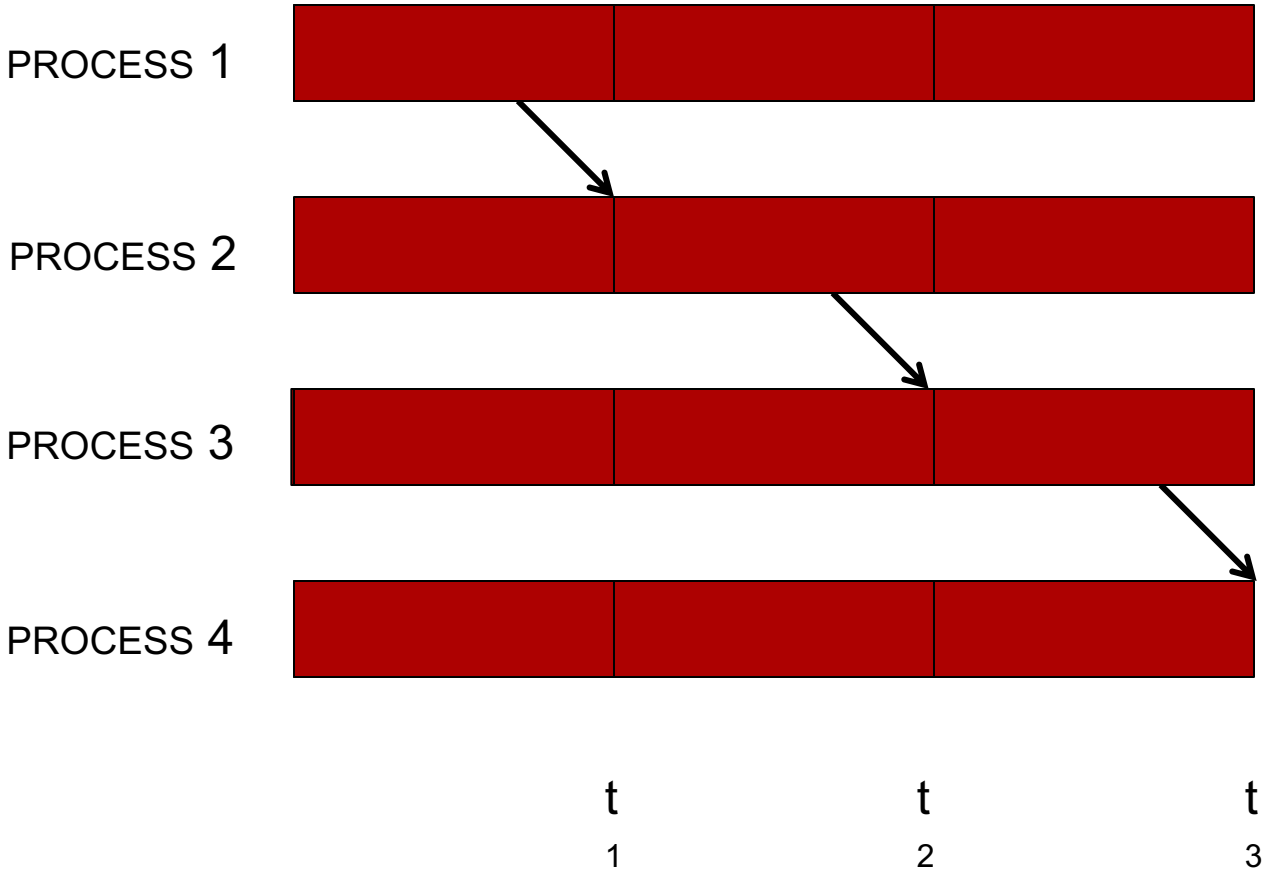
Introduction

- Today's predominant HPC resilience mechanism, *coordinated checkpoint/restart*, is unlikely to scale well
  - Coordination overheads, SDC recovery, complexity of recovery
  - Uncoordinated CR has well-examined issues as well (SC'12, SC'14, Resilience '13-'16)
- CR on future systems: more diverse, less synchronized, more CPU intensive
- Independent scheduling of resilience activity can result in delay propagation between processes

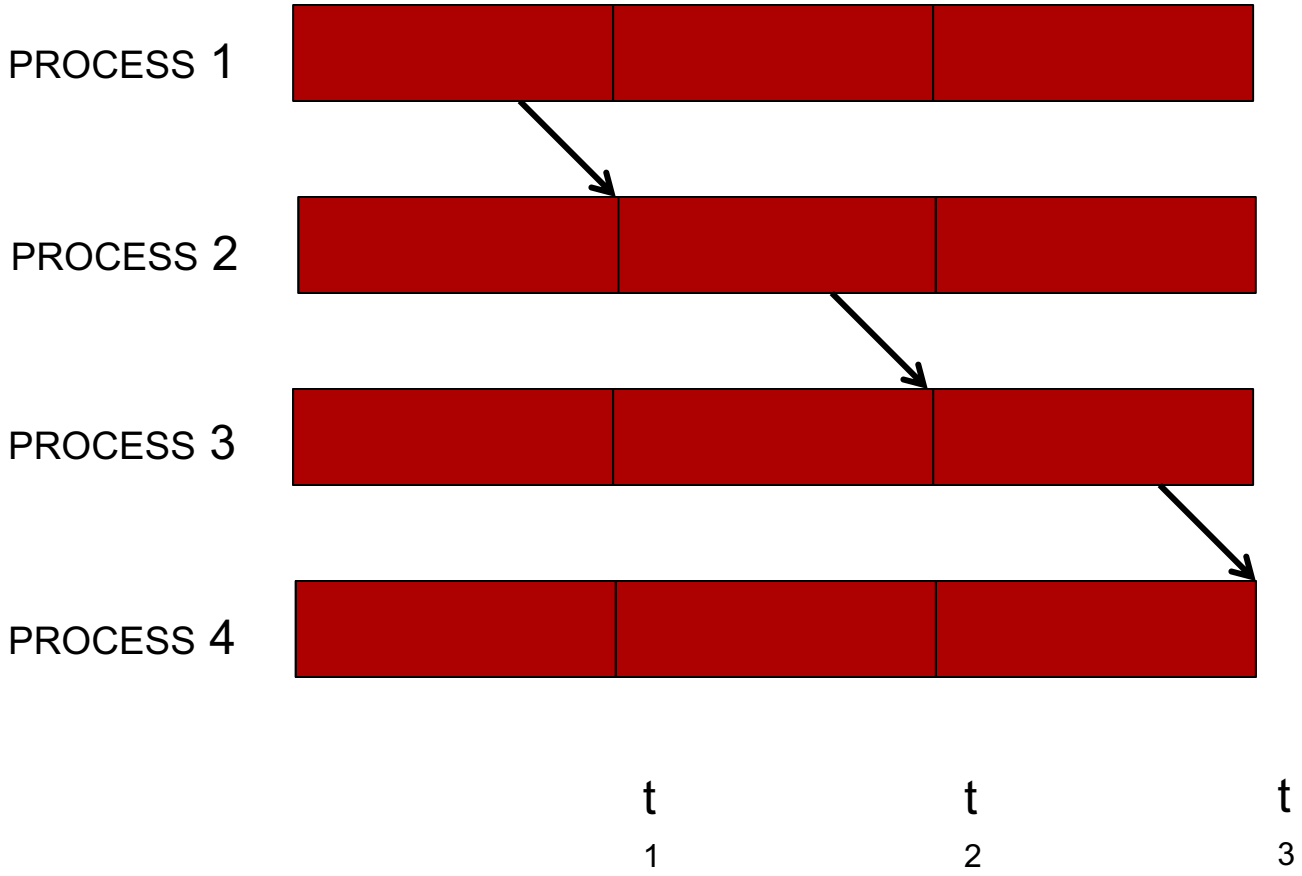
Coordinated C/R unlikely to scale



Coordinated checkpoint/restart (CCR)



Uncoordinated Checkpointing (UCR)



It’s not the heat…

- Composition (and therefore scheduling) of resilience activity will be important
  - …but few studies have considered these issues

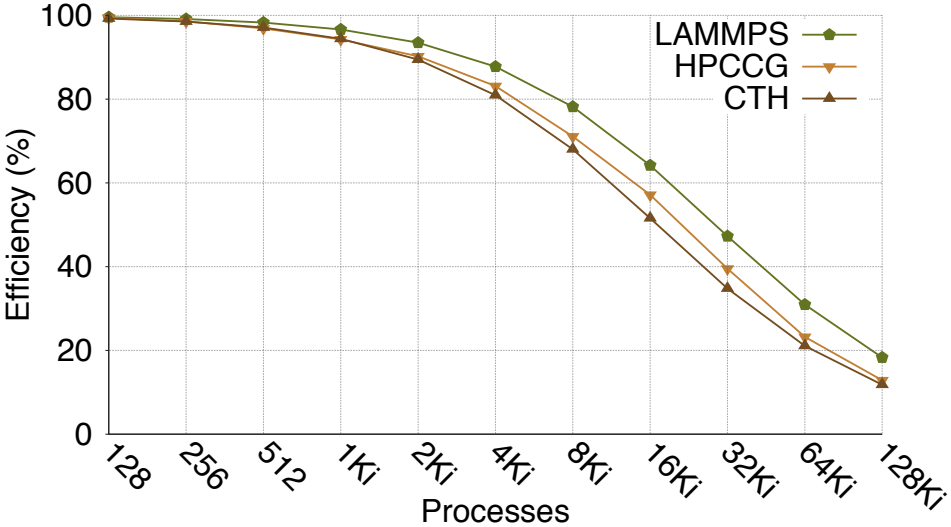
How / why do different schedules affect application performance at scale?

- Simulation-based study: scheduling subtleties can have significant performance effects
  - Aggregate time spent on resilience is not performance-predictive
  - Shorter, more frequent “resilience quanta” correlated with better scaling
  - Apps with infrequent collective operations may scale better

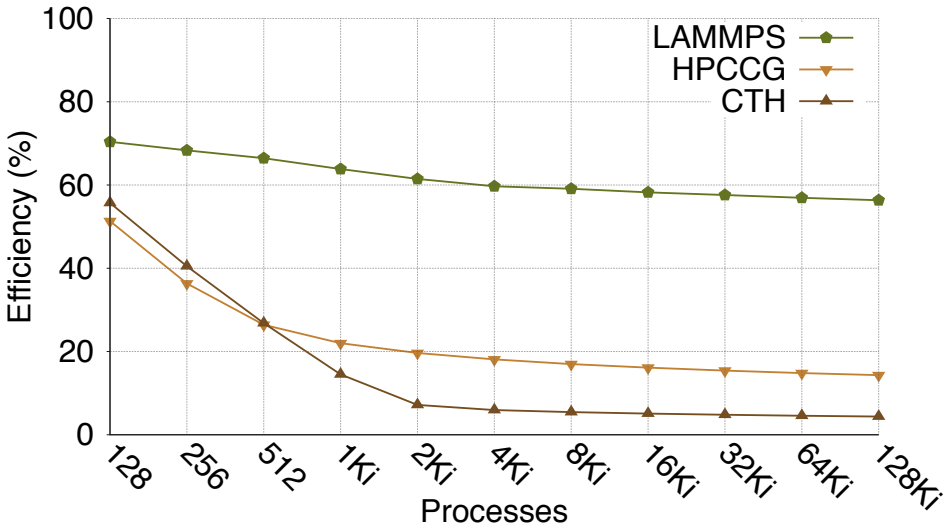
Advantages of uncoordinated checkpointing

- Each process checkpoints independently, avoids coordination overheads
- Logging of messages ensures all checkpoints are consistent
- Upon failure, only failed node(s) restarts rather than all nodes with coordinated checkpoint/restart
- Message dependencies mean restarts may propagate to other nodes

Uncoordinated overheads greater than coordinated due to analogy with "jitter"



Coordinated C/R



Uncoordinated C/R

We want to explore the space in between these extremes

Modeling different resilience schedules

- Overall resilience strategy, hardware capabilities, other factors determine resilience schedules in practice
- Simulate not just amount of activity, but its scheduling WRT application
  - Amount: different aggregate activity percentages – 1%, 5%, 10%
  - Aggregate amount of detoured CPU time over an application's execution
  - Schedule: vary duration, frequency inversely
  - Combinations chosen to explore tradeoffs rather than reflect specific examples

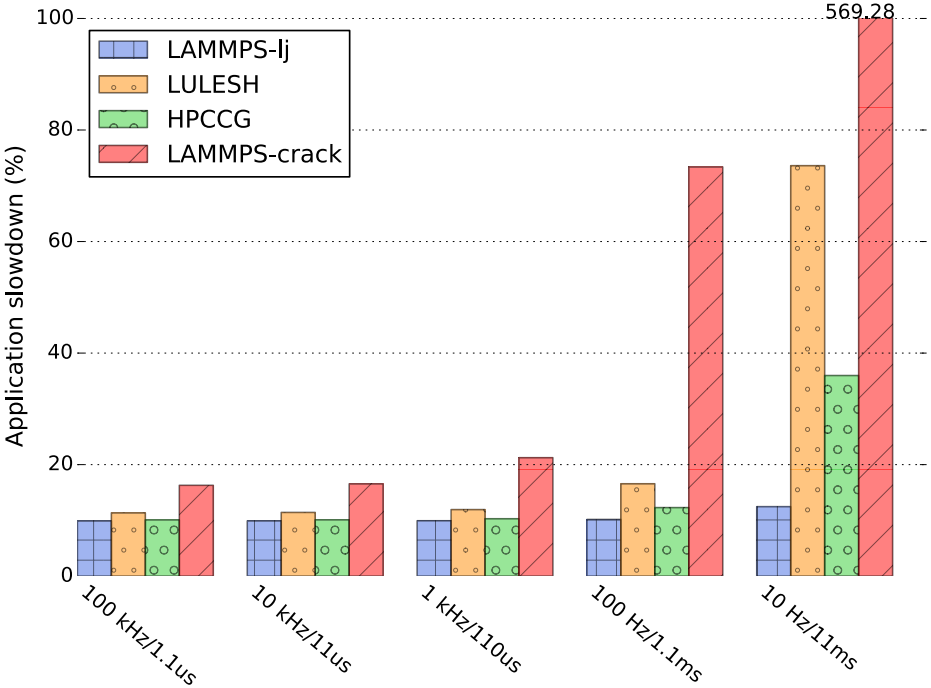
Exploration via simulation

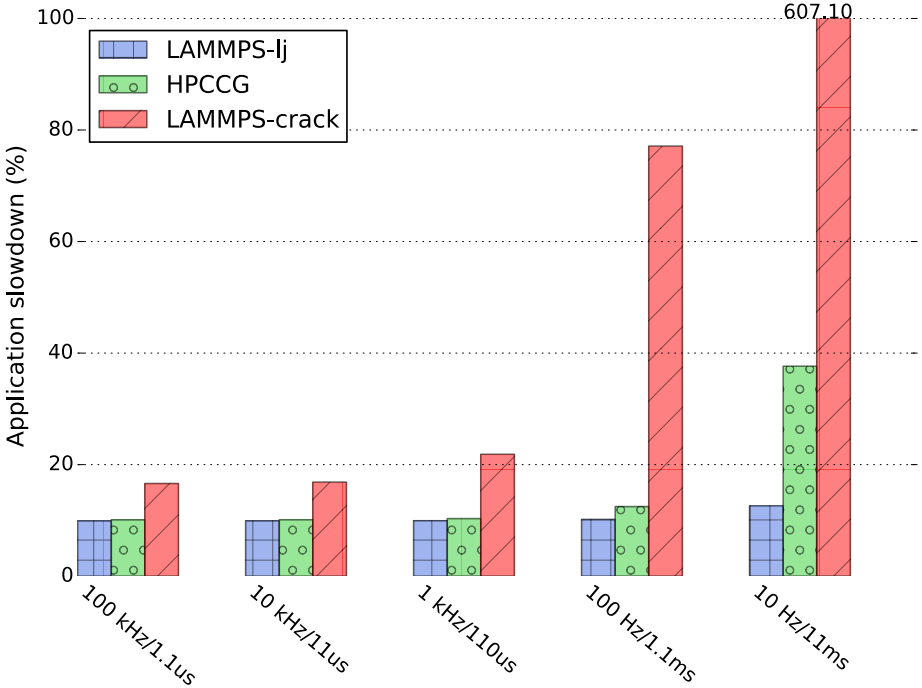
- We have a validated, well-exercised simulation framework, LogGOPSim (Levy et al. PMBS 2014)
- Checkpointing activity is modeled as CPU detours
  - Periods of time during which the CPU is taken from the application
- Simulate application behavior by replaying collected communication traces
  - All communication dependencies are reproduced
- Traces can be extrapolated from small application runs
  - Collect trace of app with p processes, extrapolate to $k * p$
  - Communication traces for MPI collectives are reproduced exactly
  - Point-to-point communications are approximated (weak scaling)
- Validated for extrapolation and prediction of local checkpointing overheads
  - Hoefler et al. SC'10, Ferreira et al. SC'14, Hoefler et al. HPDC'10

Workloads considered

- LogGOPSim with representative workload communication traces
  - LAMMPS – molecular dynamics simulation (*Lennard-Jones* and *2D-crack* potentials)
  - HPCCG – conjugate gradient solver mini-app from Mantevo
  - LULESH – shock hydrodynamics mini-app
- LogGOPSim replays the application communication trace
- Detour lists are used to introduce simulated resilience activity during trace replay
  - Separate detour list for each schedule and activity ratio
  - For each list, sum of detoured CPU time is equal to the aggregate percentage

Composition of activity has greater effect than aggregate amount

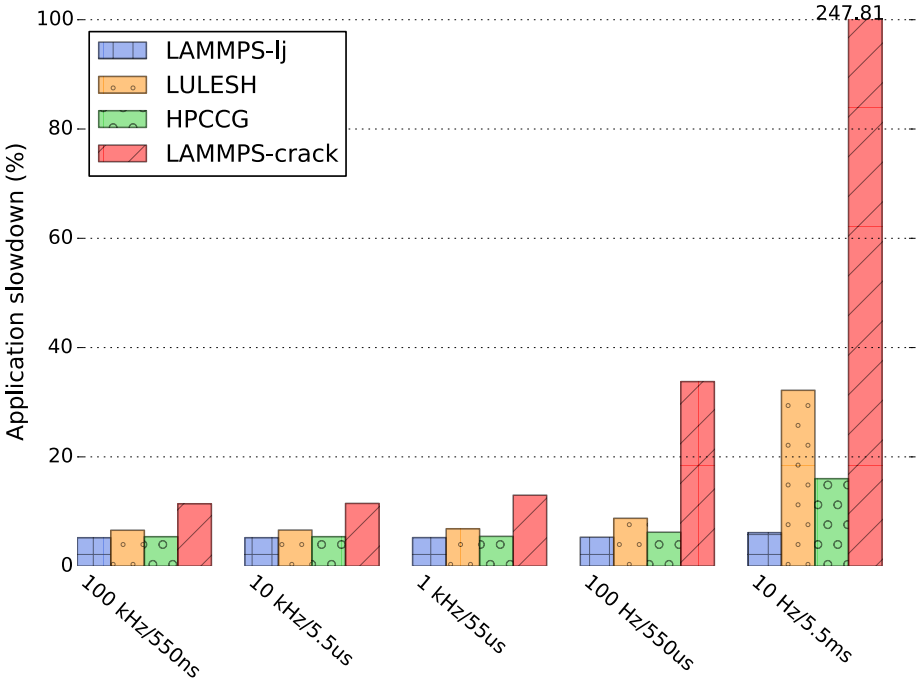


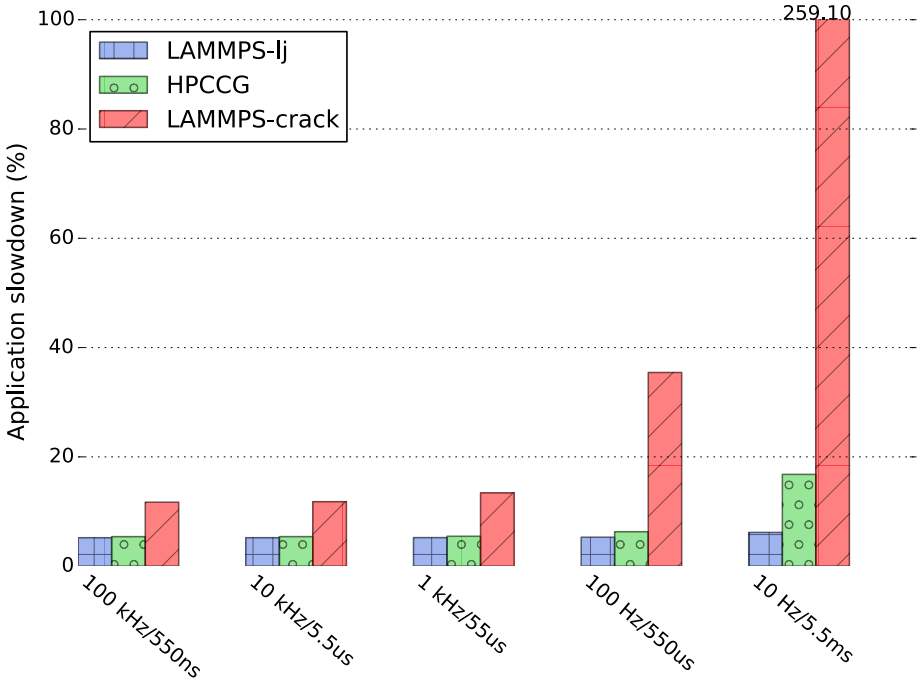


5% aggregate resilience activity

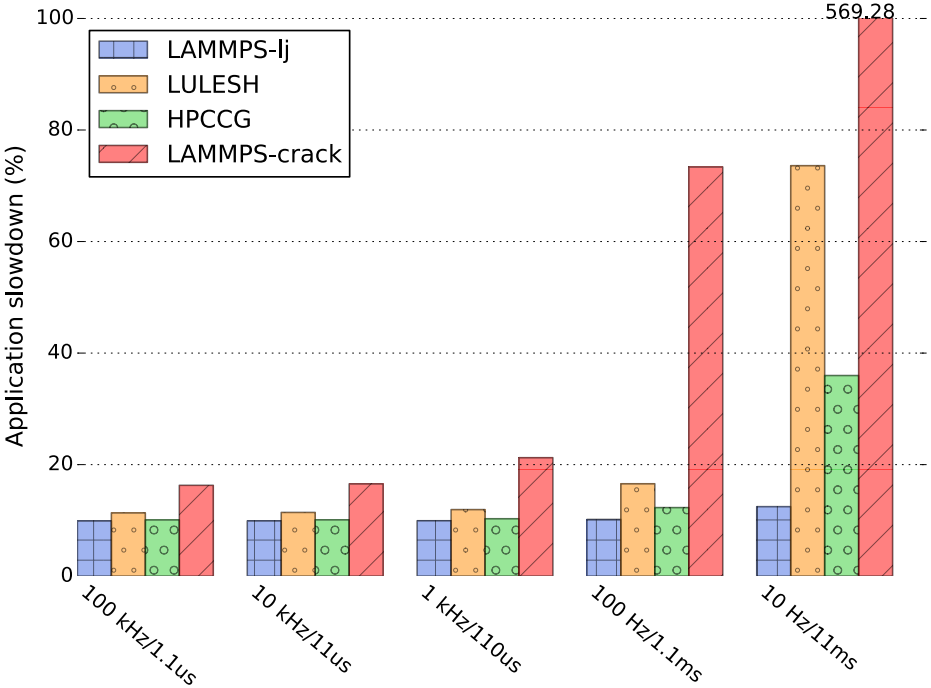
10% aggregate resilience activity

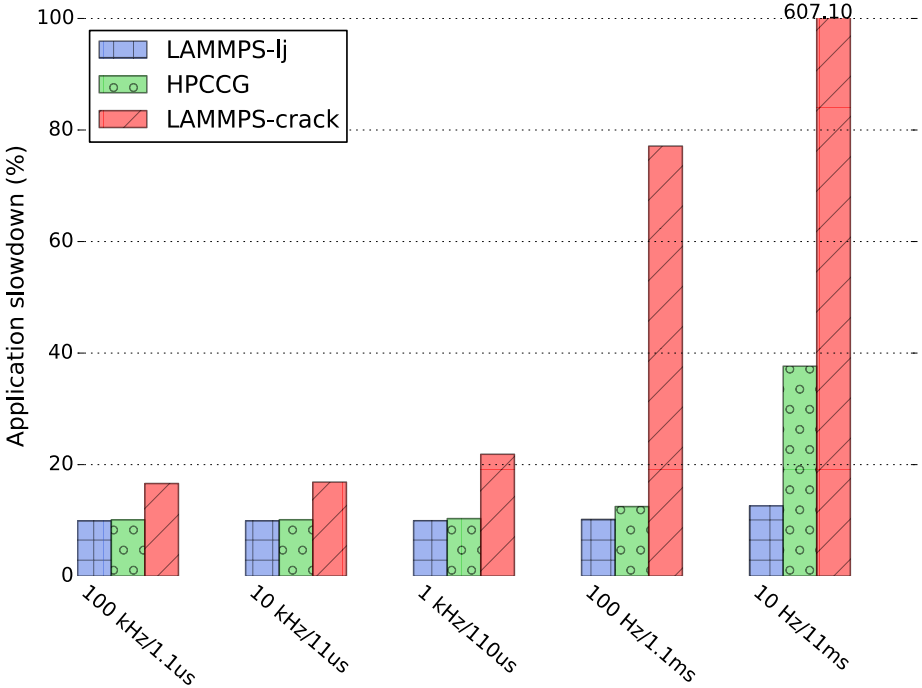
5%


247.81
LAMMPS-lj
LULESH
HPCCG
LAMMPS-crack
Application slowdown (%)
0
20
40
60
80
100
100 kHz/550ns
10 kHz/5.5us
1 kHz/55us
100 Hz/550us
10 Hz/5.5ms


259.10
LAMMPS-lj
HPCCG
LAMMPS-crack
Application slowdown (%)
0
20
40
60
80
100
100 kHz/550ns
10 kHz/5.5us
1 kHz/55us
100 Hz/550us
10 Hz/5.5ms

10%


569.28
LAMMPS-lj
LULESH
HPCCG
LAMMPS-crack
Application slowdown (%)
0
20
40
60
80
100
100 kHz/1.1us
10 kHz/11us
1 kHz/110us
100 Hz/1.1ms
10 Hz/11ms


607.10
LAMMPS-lj
HPCCG
LAMMPS-crack
Application slowdown (%)
0
20
40
60
80
100
100 kHz/1.1us
10 kHz/11us
1 kHz/110us
100 Hz/1.1ms
10 Hz/11ms

Intercollective arrival times are key

Does the composition of collectives in an application change the performance effect of different resilience activity schedules?

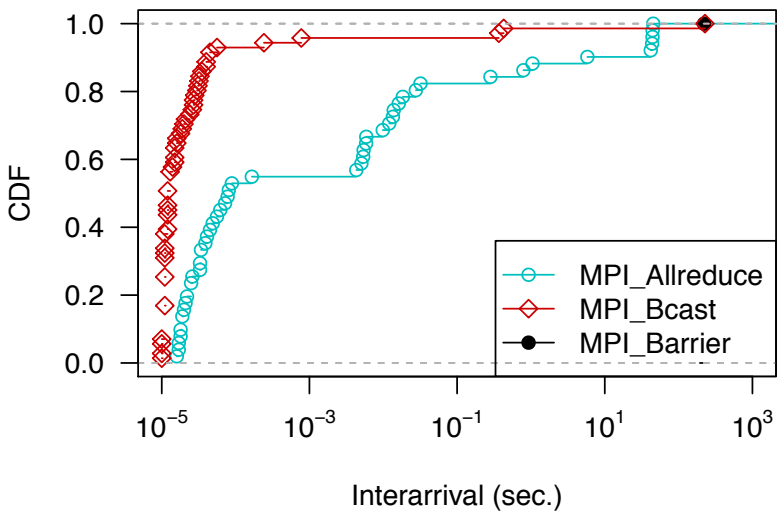
Instructive for co-design of future applications as well as for performance portability of existing applications to new systems

- Measure application collective interarrival time
- Compare against simulated slowdown results
- Our results are preliminary, but illustrate a promising analytic approach

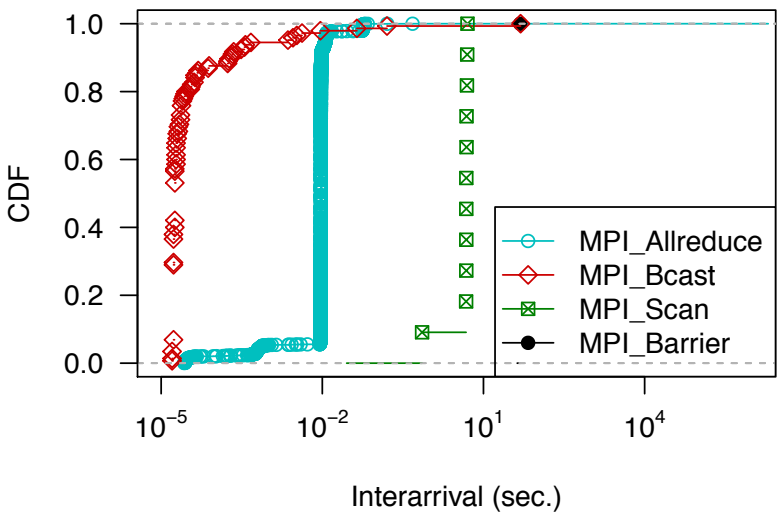
Collective interarrival in LAMMPS

Cumulative distribution functions of MPI collective interarrival time

LAMMPS L-J

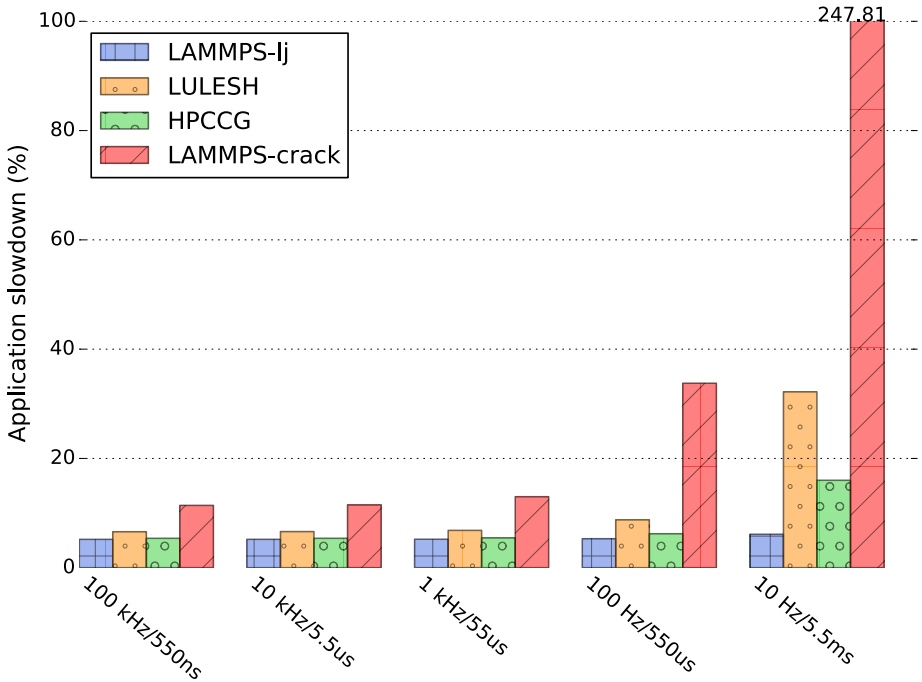


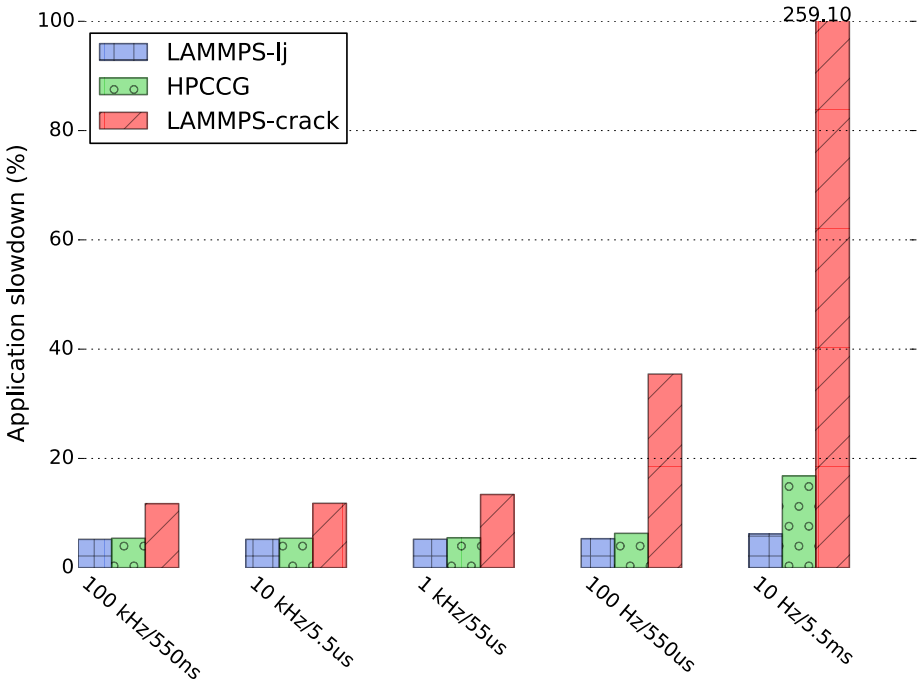
LAMMPS 2-d crack

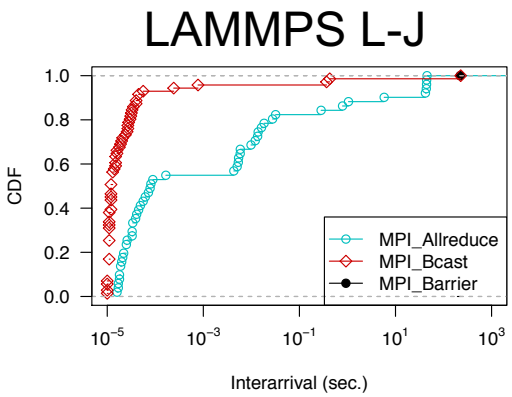


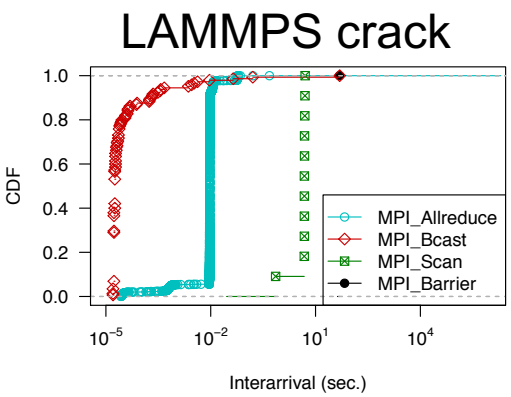
>10% of LAMMPS L-J `MPI_Allreduce()` calls have interarrival times > 5s

5% aggregate resilience activity

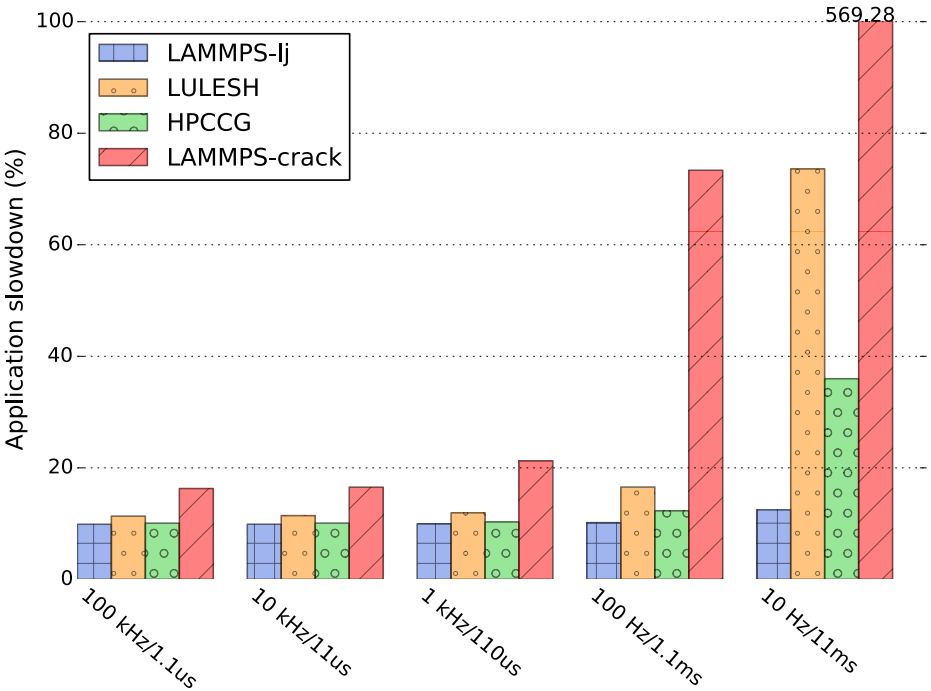


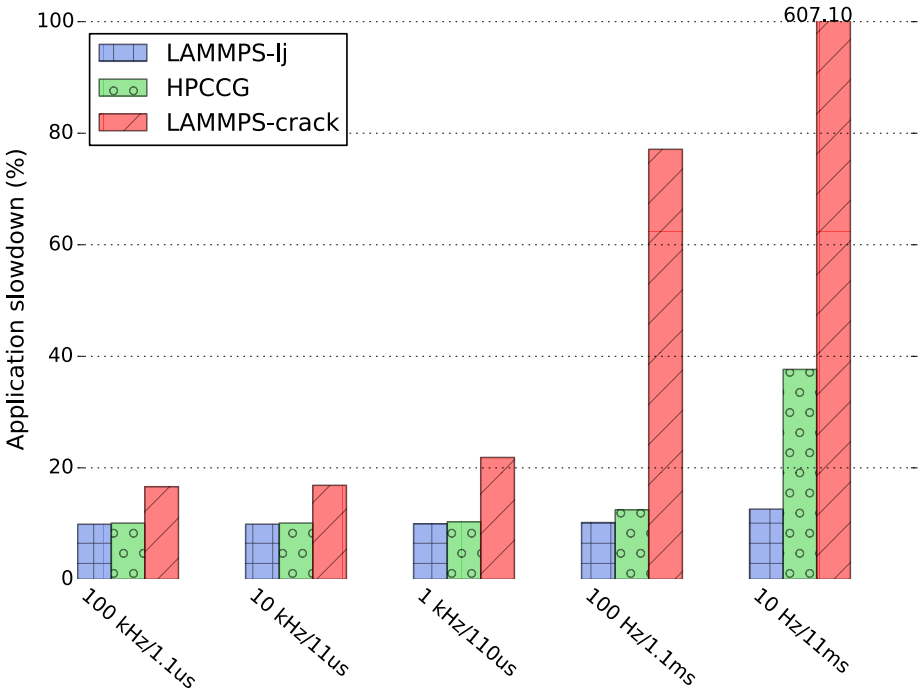




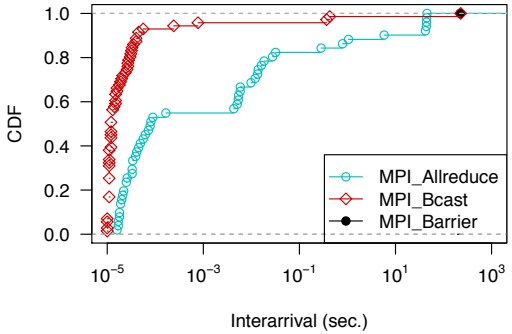


10% aggregate resilience activity

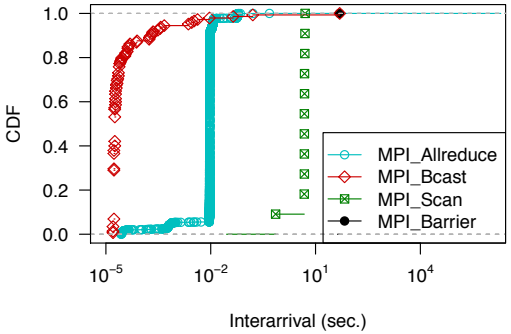




LAMMPS L-J



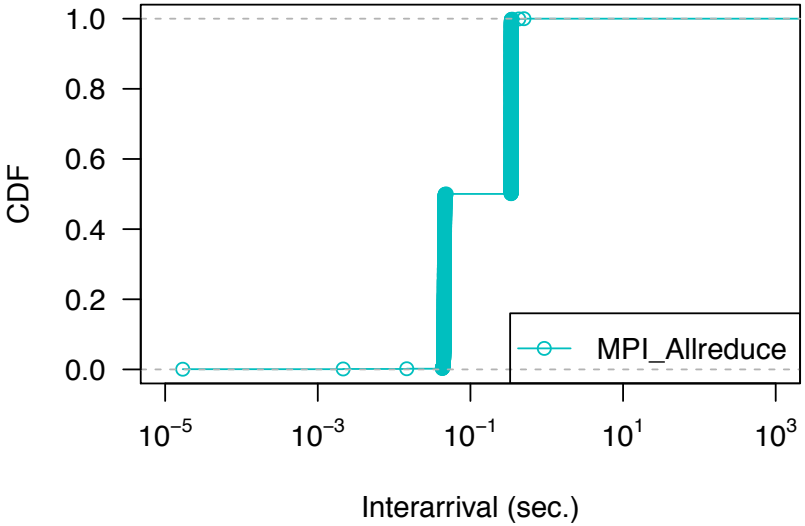
LAMMPS crack



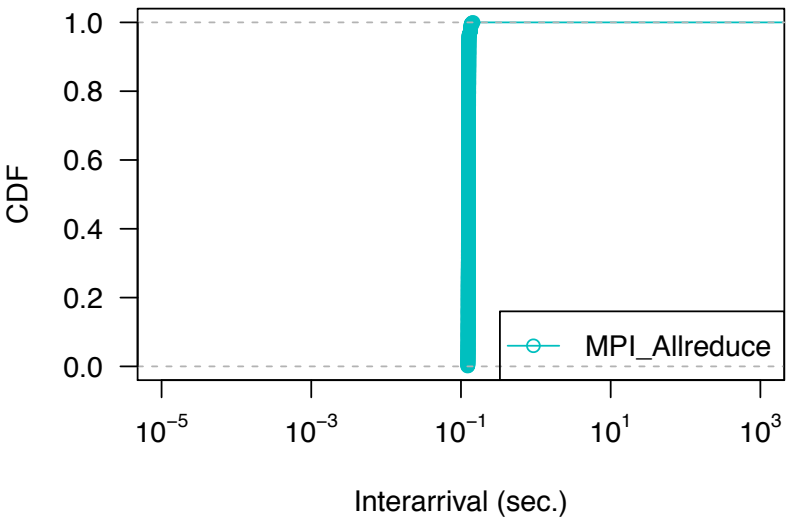
Collective interarrival in HPCCG/LULESH

Cumulative distribution functions of MPI collective interarrival time

HPCCG

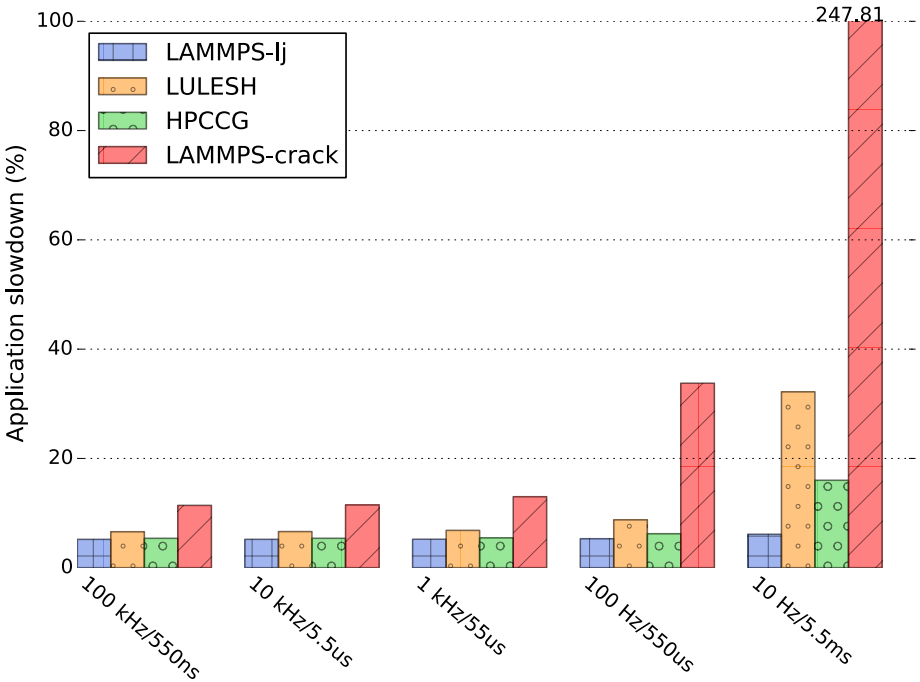


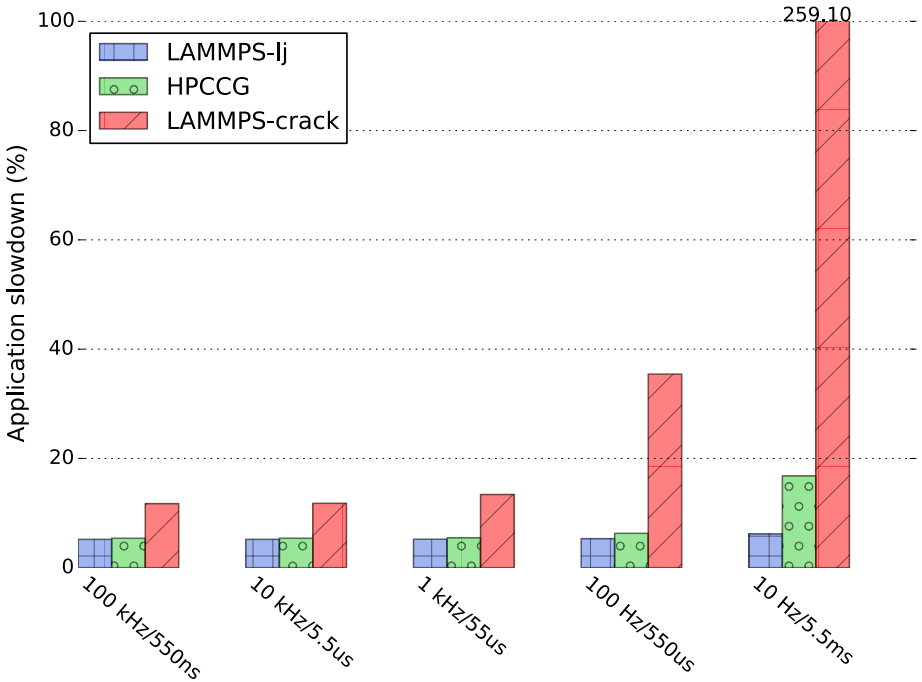
LULESH



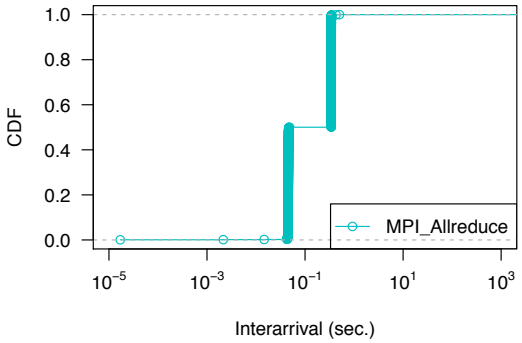
Majority of both apps' `MPI_Allreduce()` interarrivals are ~100ms

5% aggregate resilience activity

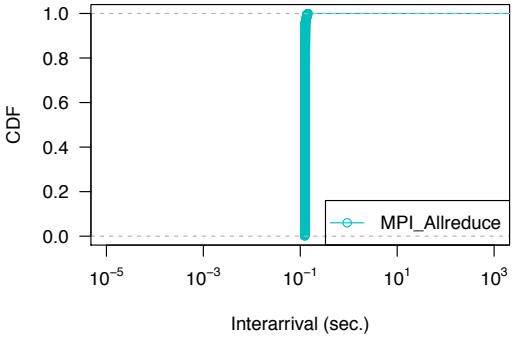




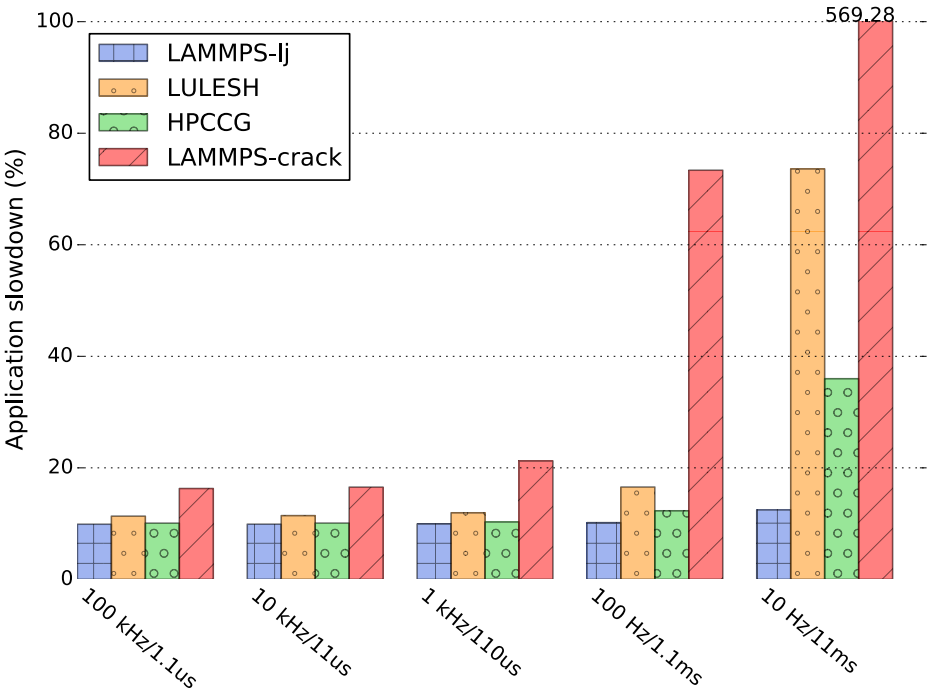
HPCCG

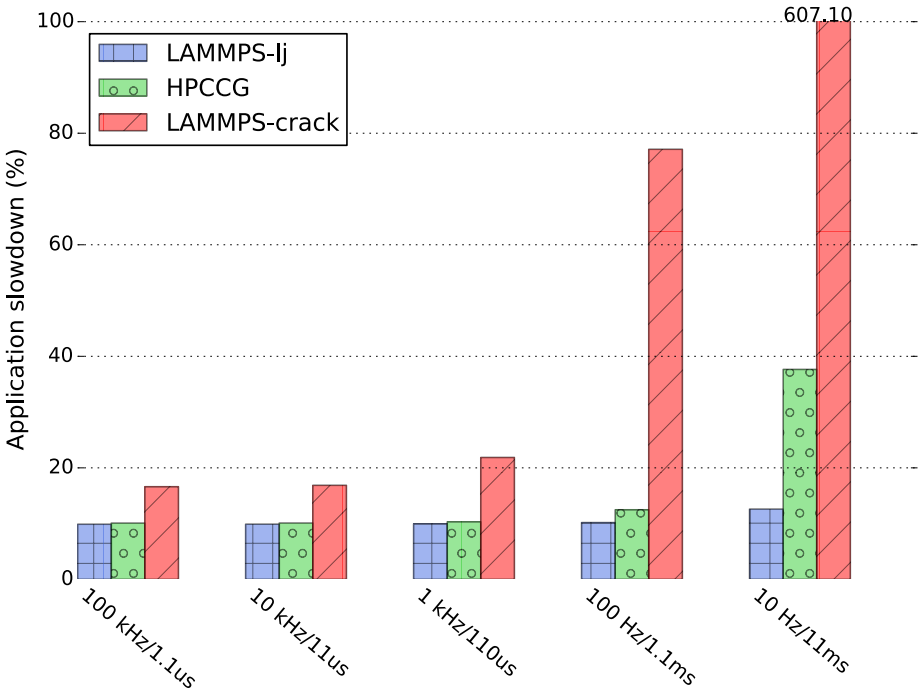


LULESH

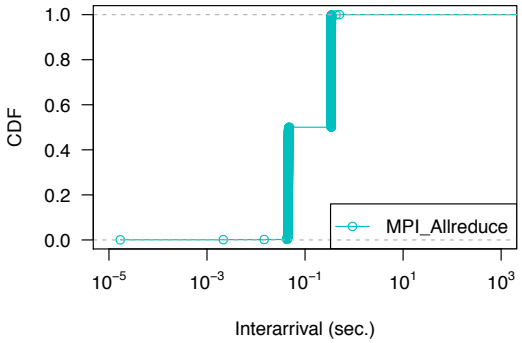


10% aggregate resilience activity

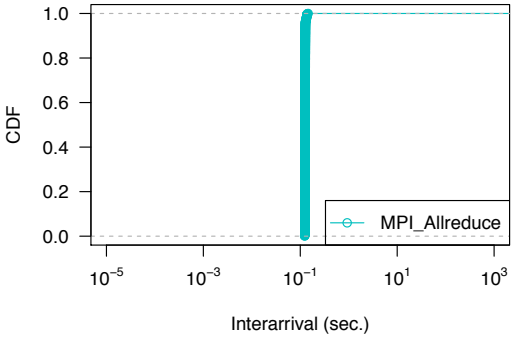




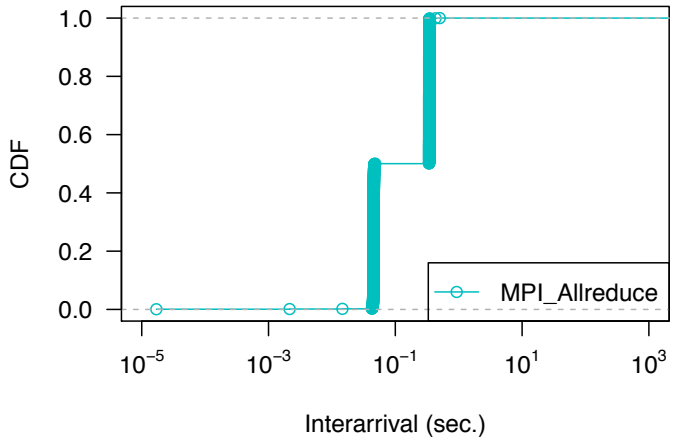
HPCCG

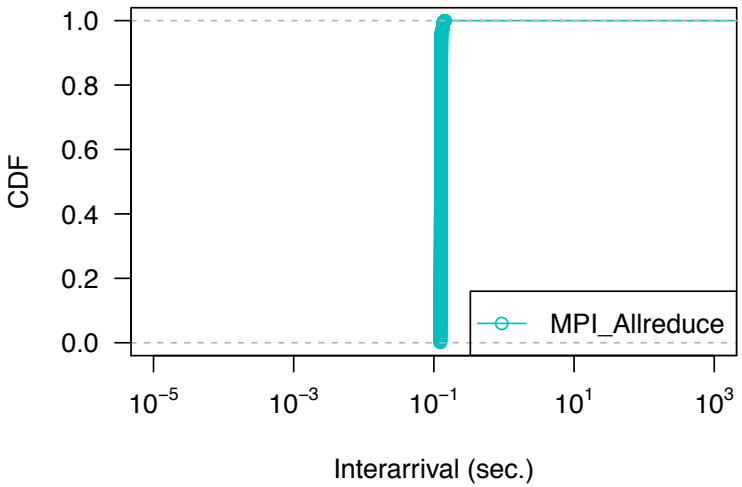


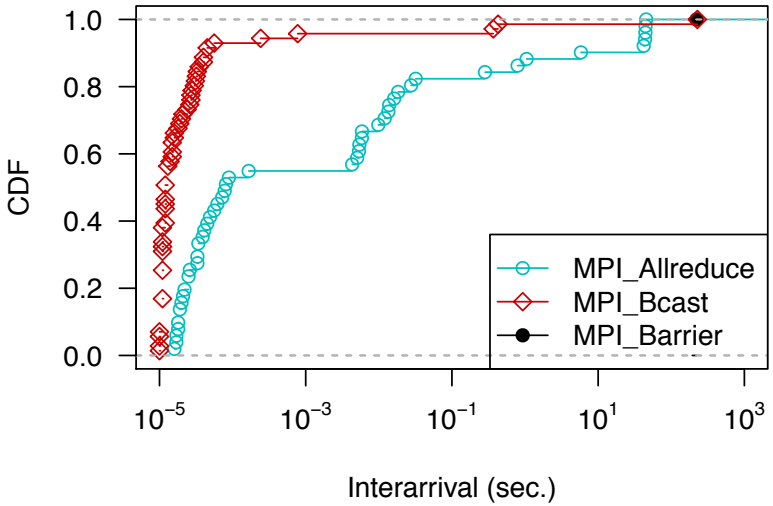
LULESH

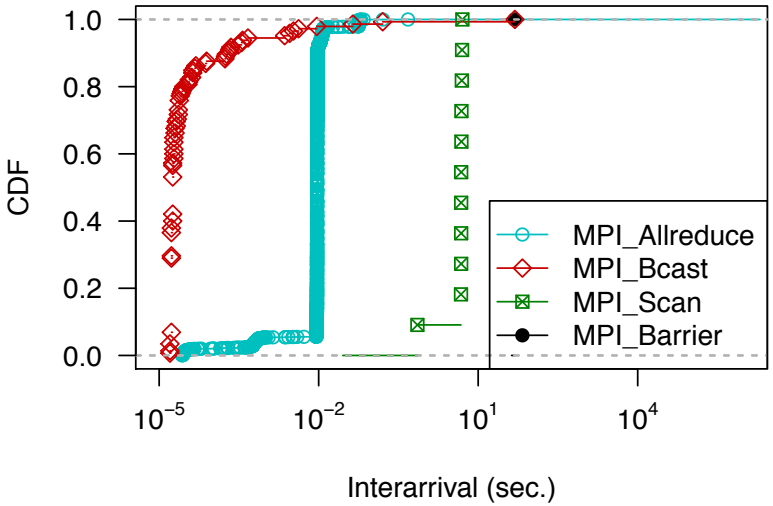


R1)HPCCG/LULESH R2)LJ/CRACK









In conclusion

- We explored the performance effects of resilience activity scheduling on HPC applications
  - Simulation technique with validated simulation infrastructure
  - Demonstrated with well-studied and important workloads
- CPU detour duration has greater impact than detour frequency
  - Reinforces earlier results from uncoordinated CR studies
- Collective interarrival time can be a factor
  - Larger proportion of longer interarrival times → less sensitive to scheduling
- This type of simulation approach will be valuable for exascale application co-design
- Acknowledgements
  - Support: DOE ASC