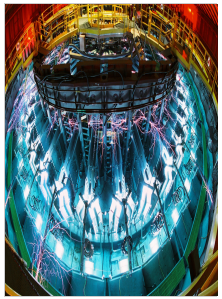


Sandia National Laboratories

Exceptional service in the national interest

Horseshoes & hand grenades (& coordination)

The case for approximate coordination in local checkpointing protocols

Patrick M. Widener, Kurt Ferreira, Scott Levy

Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, USA

patrick.widener@sandia.gov

U.S. DEPARTMENT OF ENERGY

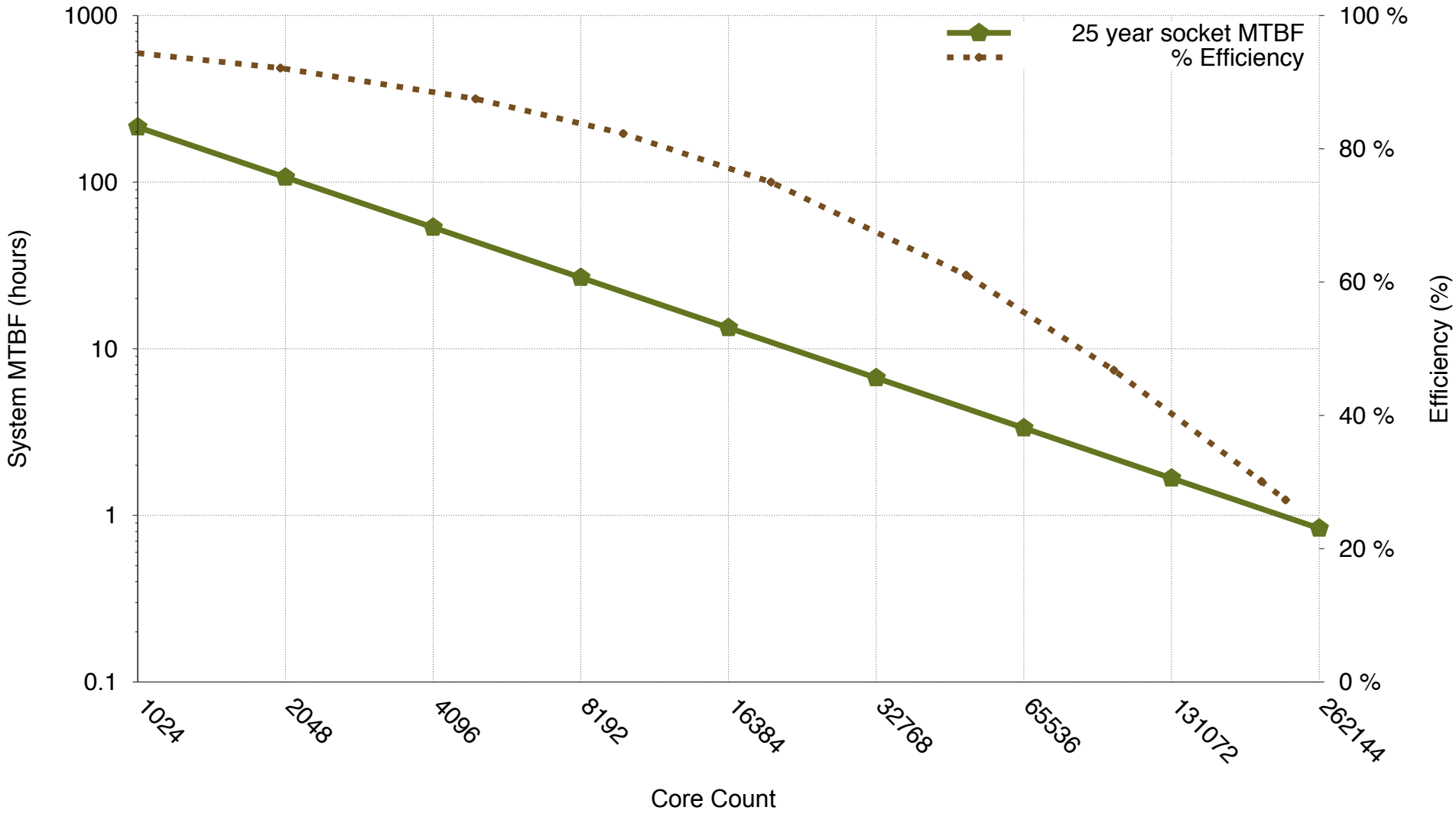
NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND2016-5027C

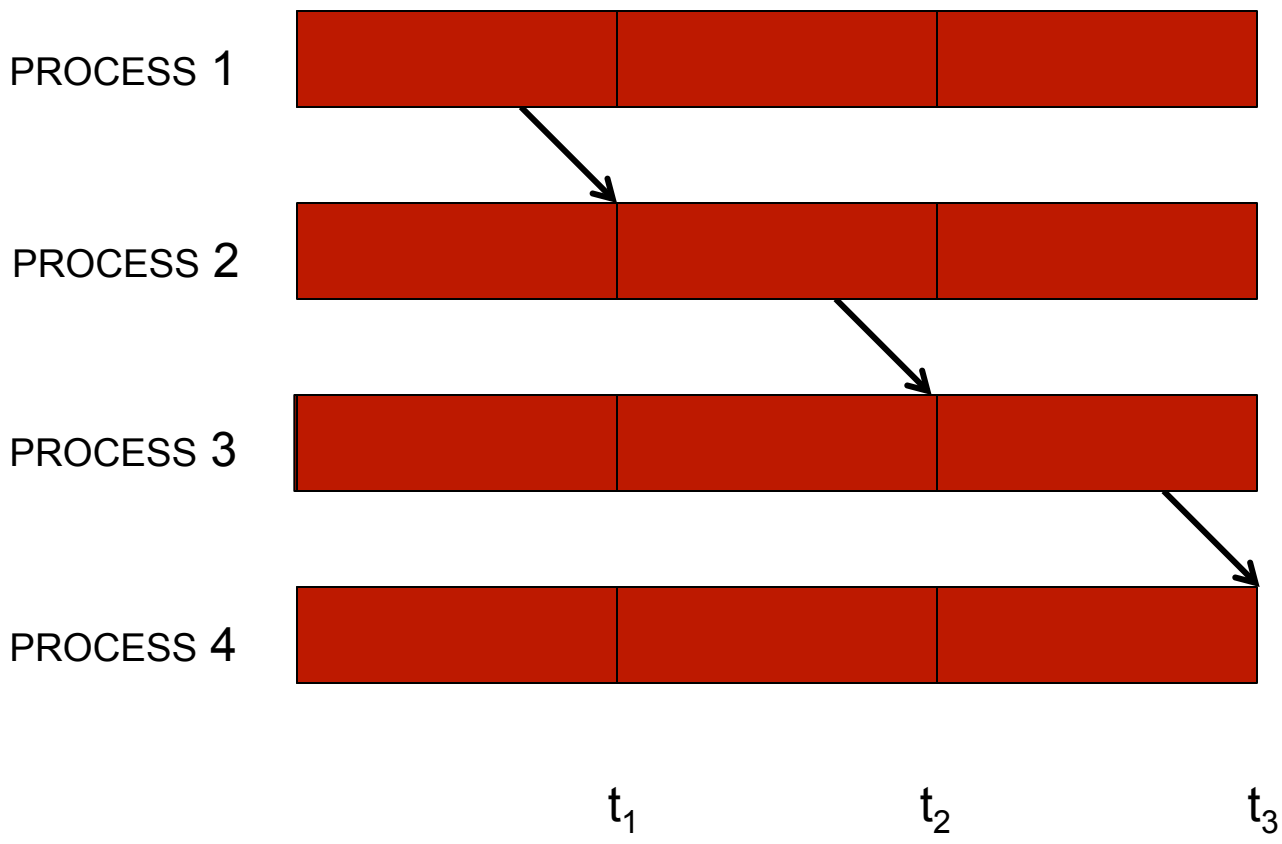
Overview

- Today's predominant HPC resilience mechanism, *coordinated checkpoint/restart*, is unlikely to scale well
  - Resource contention introduced by coordination
- Uncoordinated solutions have been proposed, but also are unlikely to scale well
  - Communication-induced delays propagate between processes (Ferreira et al. SC'14)
- Our simulation-based exploration suggests that there is a "sweet spot" between the two coordination extremes
  - We call this *approximate coordination*
  - Characterizes the tradeoff between relieving resource contention and avoiding delay propagation
  - Achievable at reasonable cost

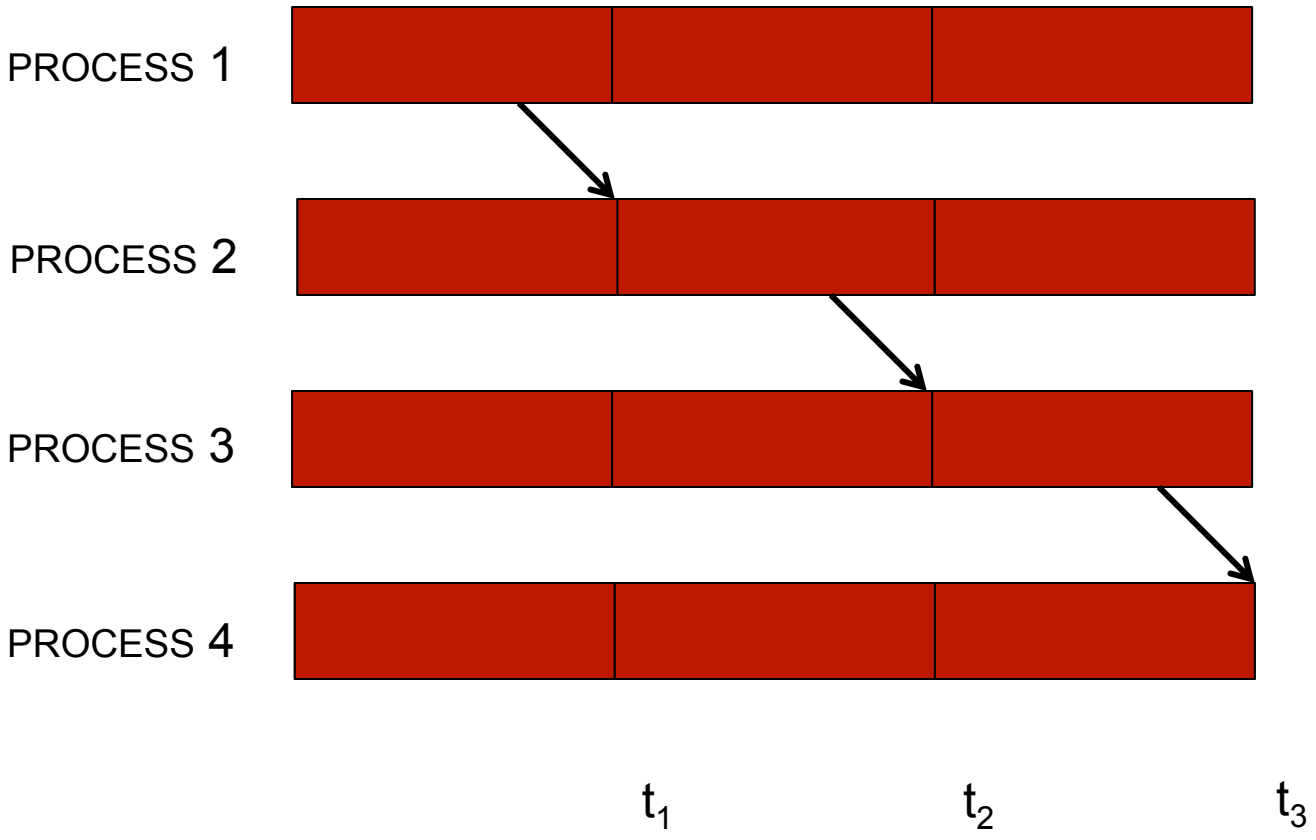
Coordinated C/R unlikely to scale



Coordinated checkpoint/restart (CCR)



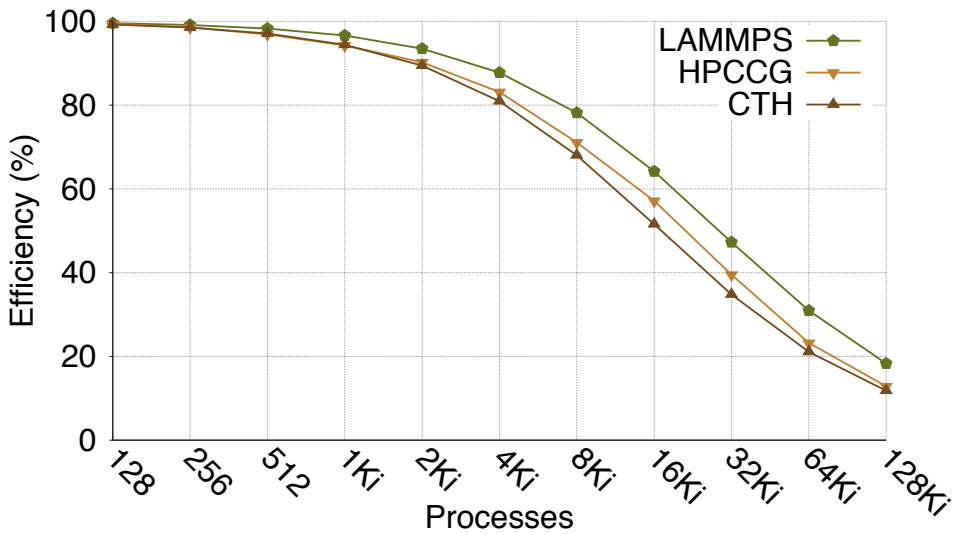
Uncoordinated Checkpointing (UCR)



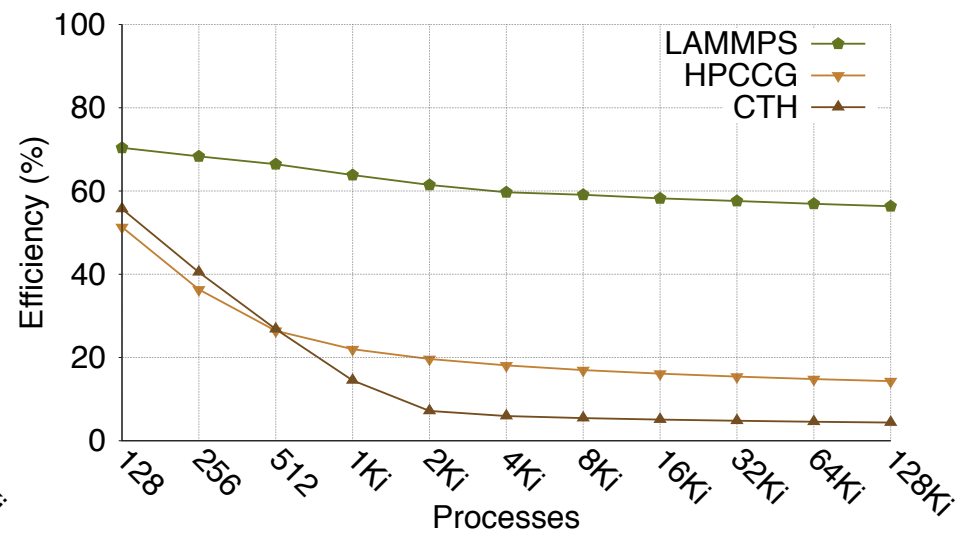
Advantages of uncoordinated checkpointing

- 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

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

Exploration via simulation

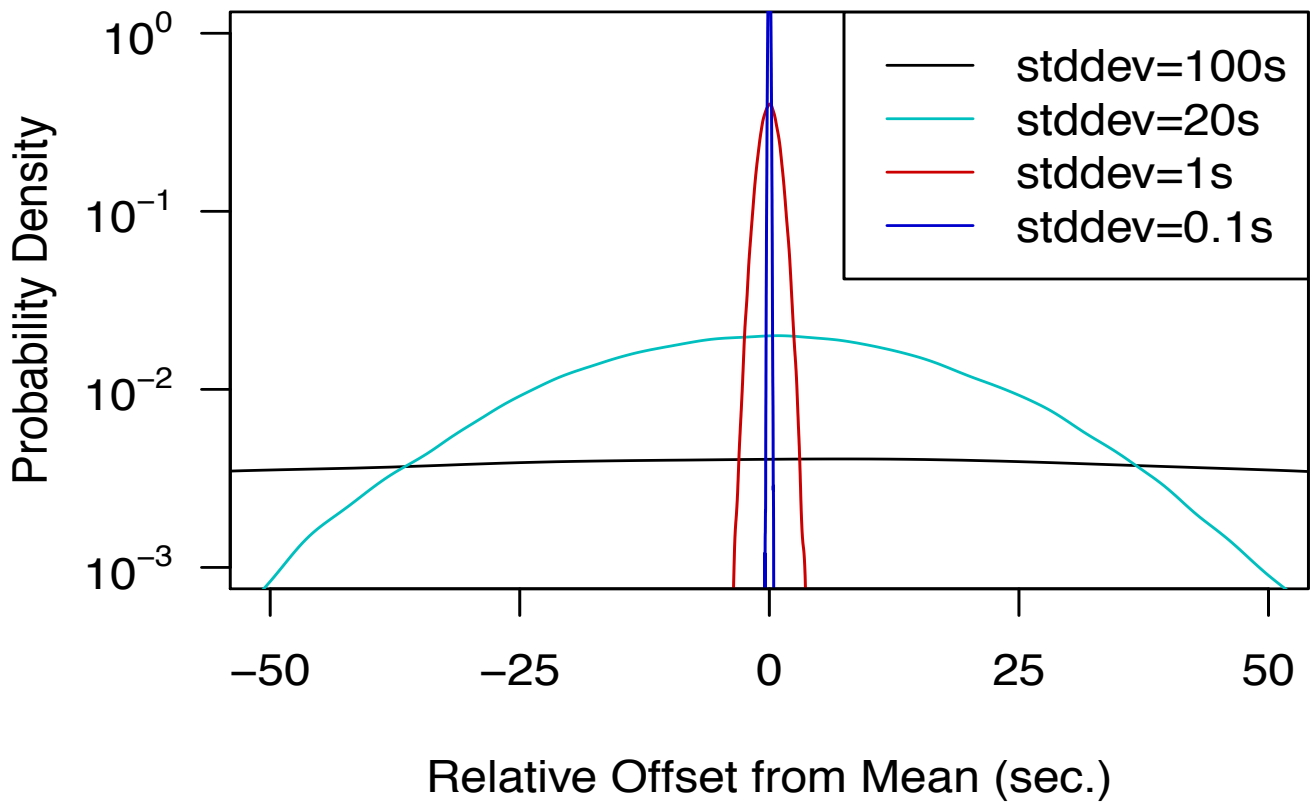
- We have a validated, well-exercised simulation framework, LogGOPSim (Levy et al. PMBS 2014)
- Checkpointing activity is modeled as CPU detours
  - Periods of time during which the CPU is taken from the application
- Simulate application behavior by replaying collected communication traces
  - All communication 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
- Validated for extrapolation and prediction of local checkpointing overheads
  - Hoefler et al. SC'10, Ferreira et al. SC'14, Hoefler et al. HPDC'10

Effect of different levels of approximate coordination

- We simulated executions of a set of workloads with varying degrees of checkpointing coordination
  - LAMMPS – molecular dynamics simulation
  - CTH – large material deformation / strong shock modeling
  - HPCCG – conjugate gradient solver mini-app from Mantevo
  - LULESH – shock hydrodynamics mini-app
- Vary the standard deviation of the normal distribution used to produce starting offsets
- Measure application slowdown in each case, for different process counts

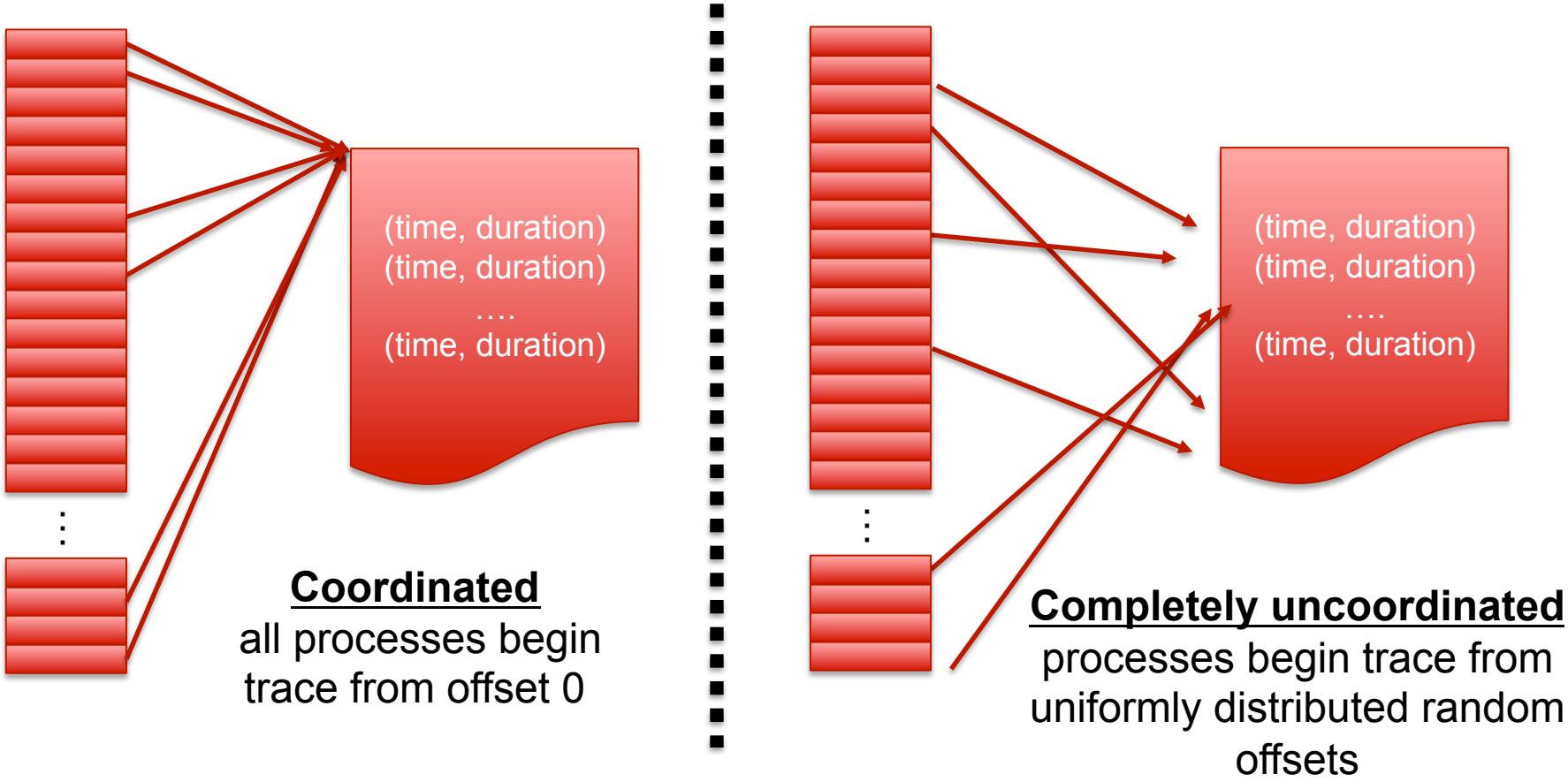
Varying the degree of coordination

Drawing offsets from a normal distribution and varying the standard deviation simulates different degrees of coordination

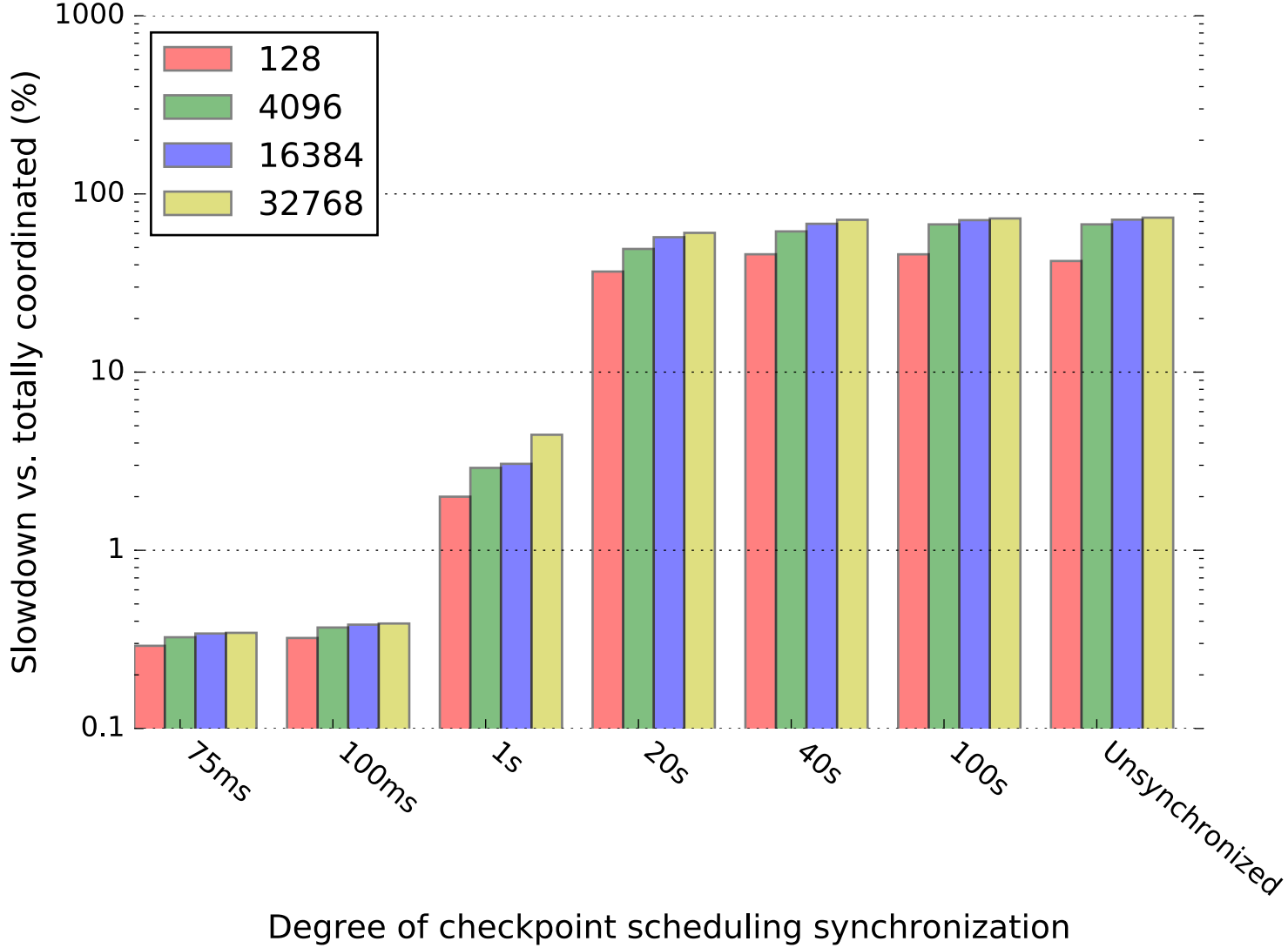


Simulating the role of coordination

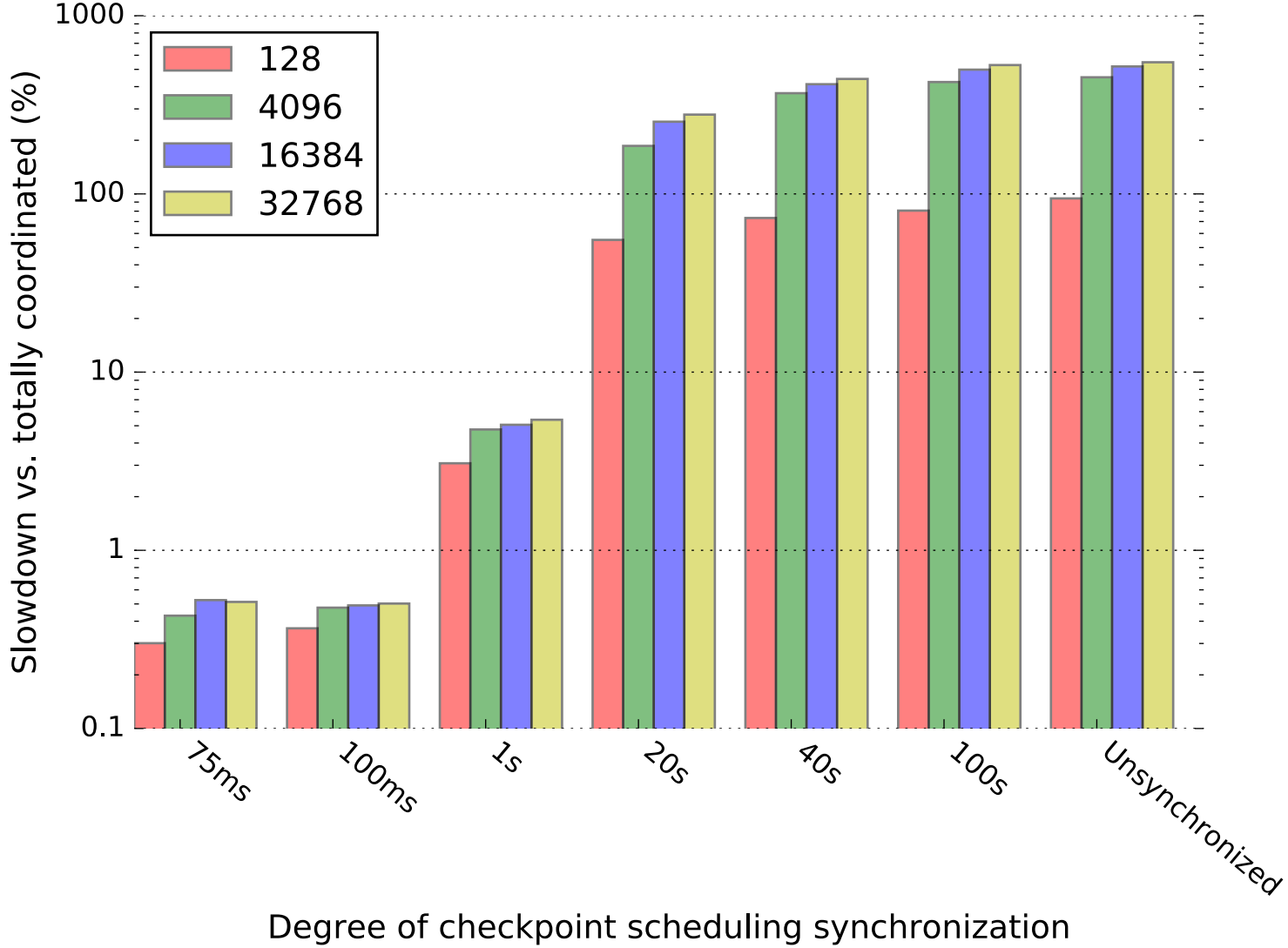
- LogGOPSim uses a *detour trace* to represent checkpoint events



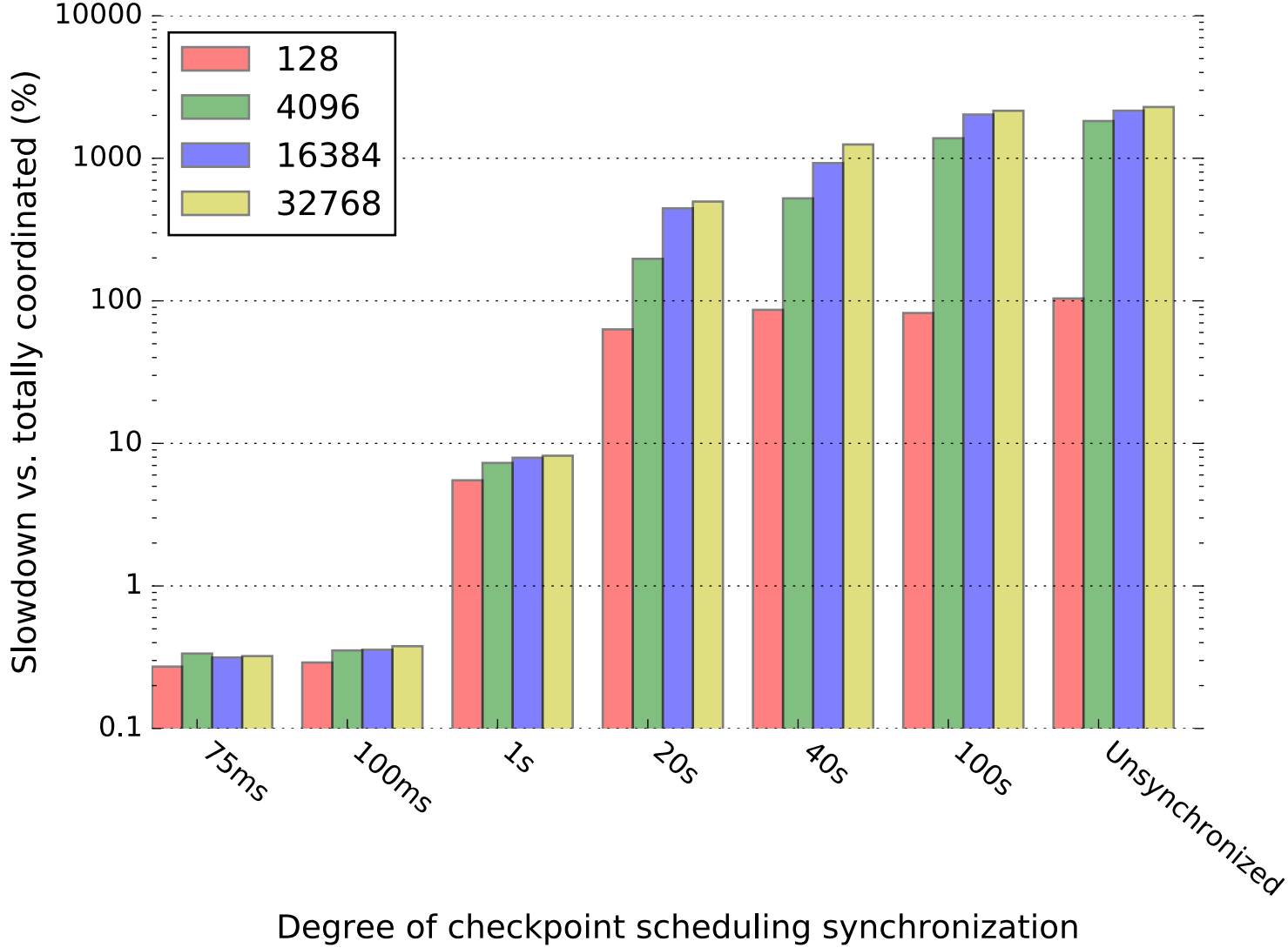
LAMMPS/LJ


1000
100
10
1
0.1
128
4096
16384
32768
Slowdown vs. totally coordinated (%)
75ms
100ms
1s
20s
40s
100s
Unsynchronized
Degree of checkpoint scheduling synchronization

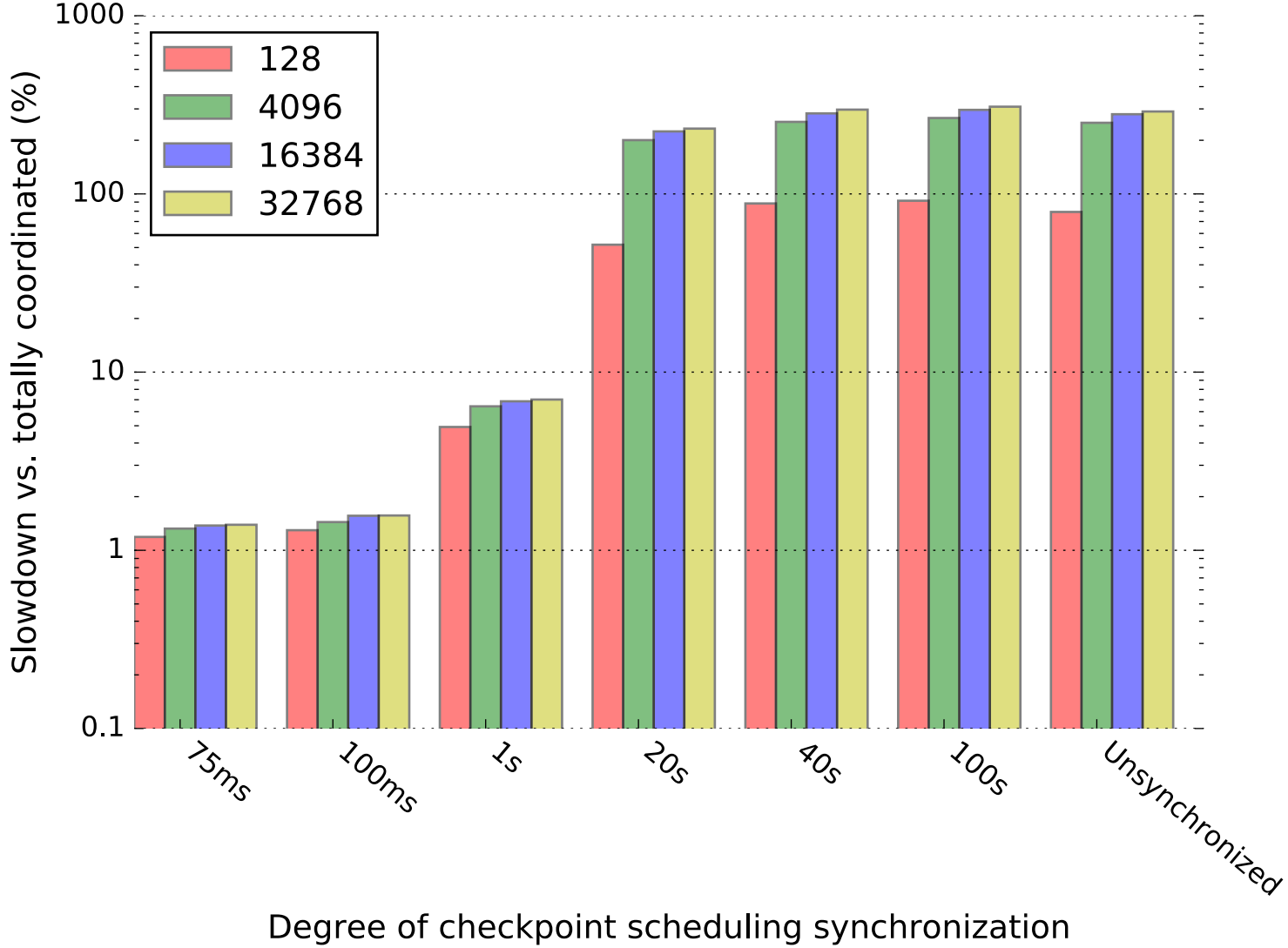
HPCCG


1000
100
10
1
0.1
128
4096
16384
32768
Slowdown vs. totally coordinated (%)
75ms
100ms
1s
20s
40s
100s
Unsynchronized
Degree of checkpoint scheduling synchronization

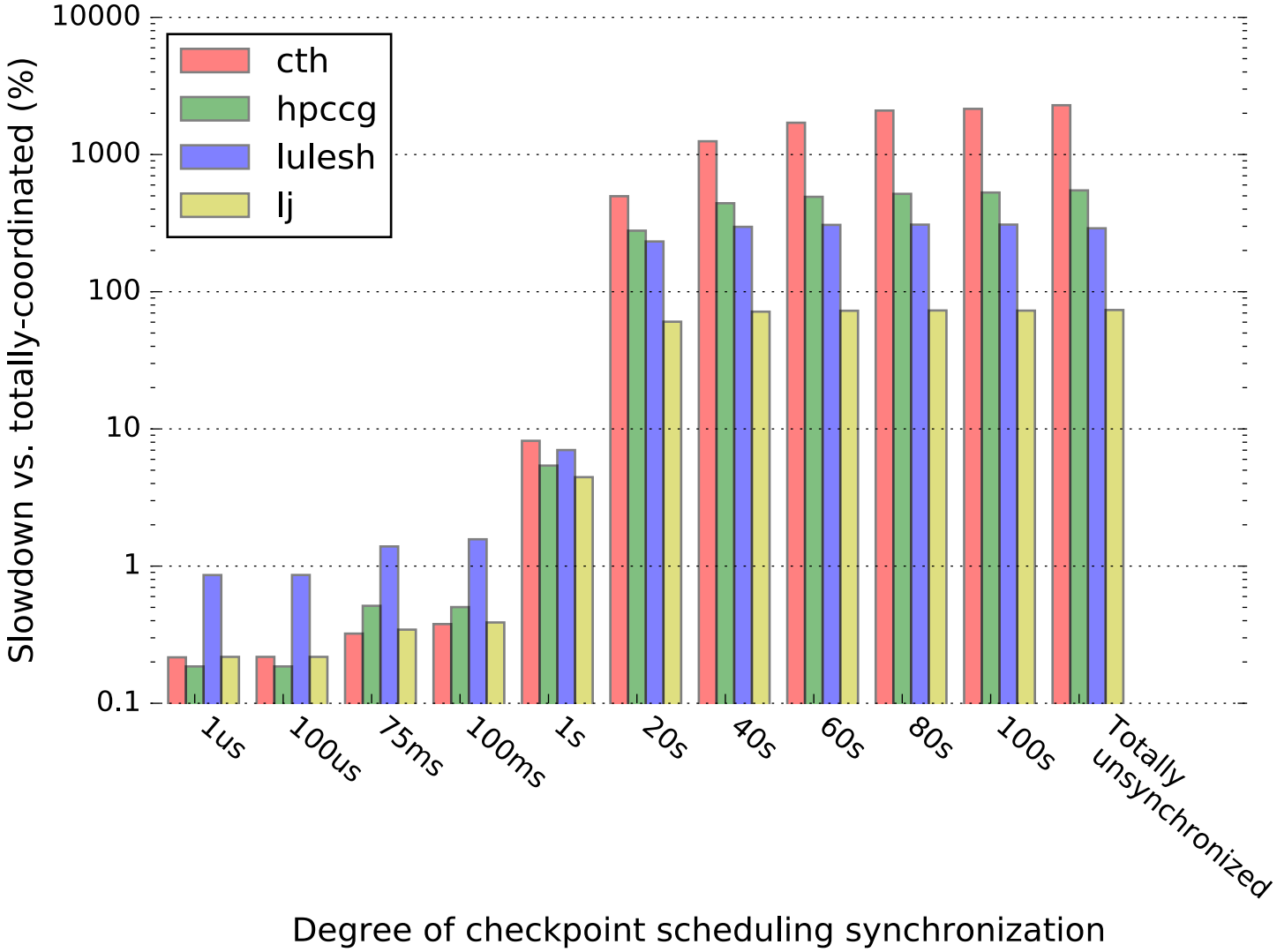
CTH


10000
1000
100
10
1
0.1
128
4096
16384
32768
Slowdown vs. totally coordinated (%)
75ms
100ms
1s
20s
40s
100s
Unsynchronized
Degree of checkpoint scheduling synchronization

LULESH


1000
100
10
1
0.1
128
4096
16384
32768
Slowdown vs. totally coordinated (%)
75ms
100ms
1s
20s
40s
100s
Unsynchronized
Degree of checkpoint scheduling synchronization

All applications at 32Ki processes



Simulation indicates relatively loose coordination is sufficient

- For normally distributed offsets with stddev=100ms
  - On average 95% of checkpoints happen within a 200ms window
  - Application runtime increases less than 5%
  - This is well within the capabilities of systems with hardware support
    - Dedicated global interconnects or GPS cards have achieved skews ~ 1 us
- Even extremely loose coordination is viable
  - Approximately 10% slowdown for our studied workloads
  - Easily realizable in modern HPC systems
  - Also possible with acceptable reliabliity in wide-area/cloud contexts
- Diminishing returns – tighter synchronization does not result in improved performance
- Insensitive to scale of application

In conclusion

- We have studied *approximate coordination* in uncoordinated checkpointing
  - Simulation technique with validated simulation infrastructure
  - Demonstrated with well-studied and important workloads
- This type of simulation approach will be valuable for exascale application co-design
- Approximate synchronization appears viable
  - Even modest coordination levels give acceptable performance
  - Dedicated hardware is not necessary
  - And you may get it for free depending on your collective operations (EuroMPI 2016)
- Acknowledgements
  - Support: DOE ASC