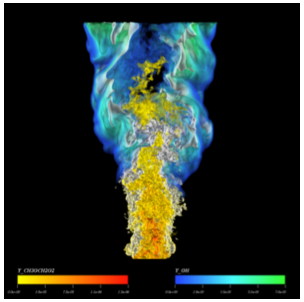


Sandia National Laboratories

Exceptional service in the national interest

How I Learned to Stop Worrying and Love In Situ Analytics

Leveraging Latent Synchronization in MPI Collective Algorithms

Scott Levy, Kurt B. Ferreira, Patrick Widener

Center for Computing Research

Sandia National Laboratories

Patrick G. Bridges, Oscar H. Mondragon

Department of Computer Science

University of New Mexico

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

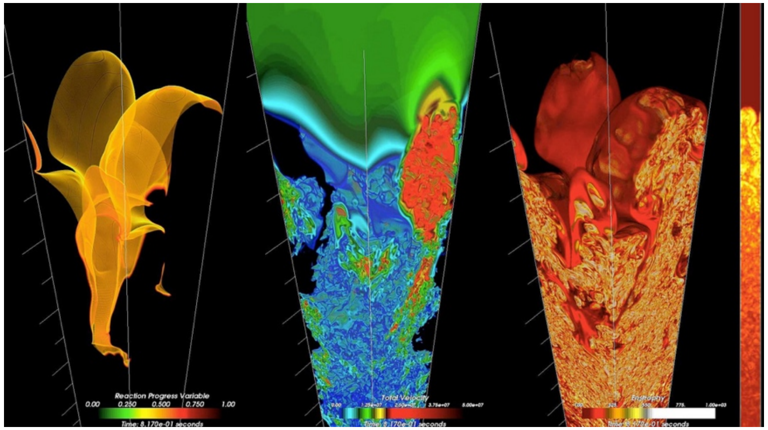
SAND2016-9184C

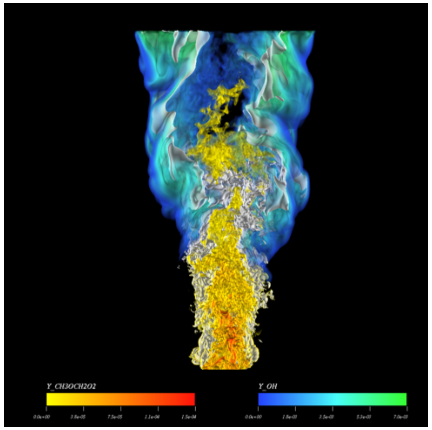
CCR Center for Computing Research

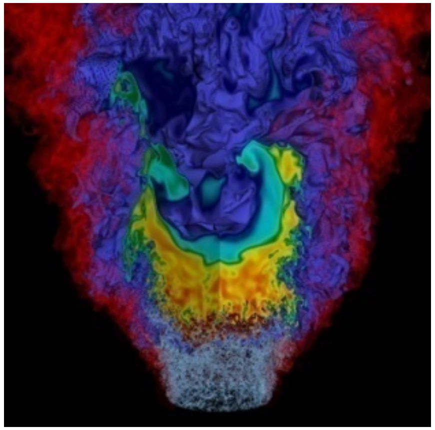
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Why Analytics?

- Scientific simulations generate terabytes of output data
- Processing allows domain scientists to more easily reason about simulation results
- Common examples of data analysis
  - visualization
  - feature extraction
  - summary statistics



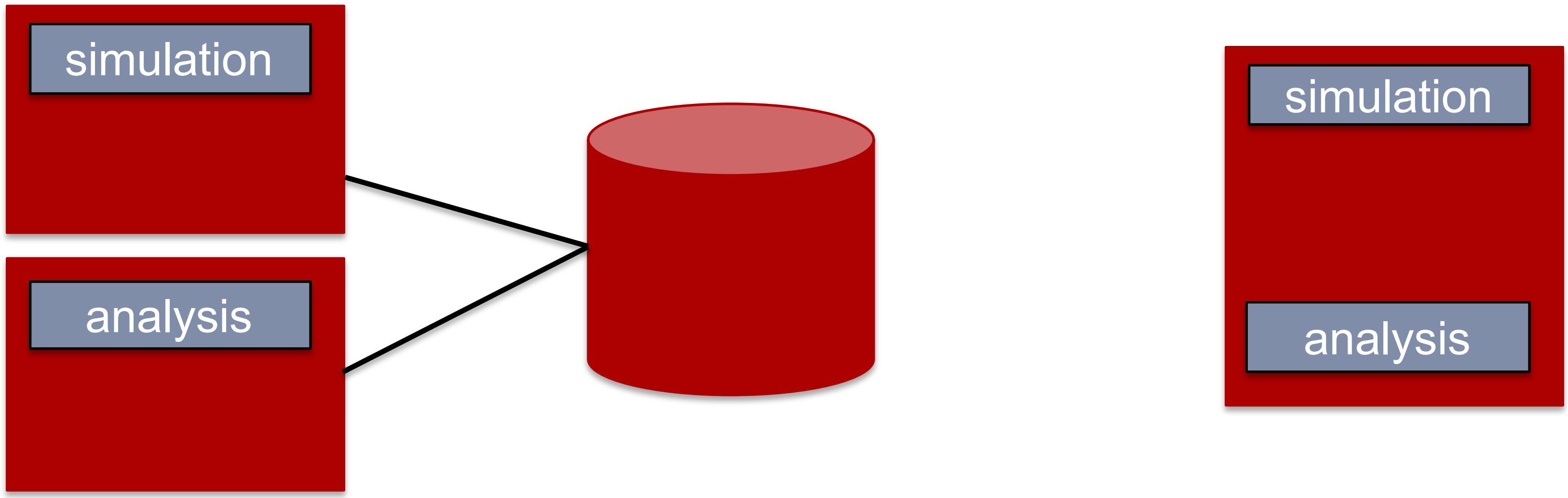




Why In Situ Analytics?

Sandia National Laboratories

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.

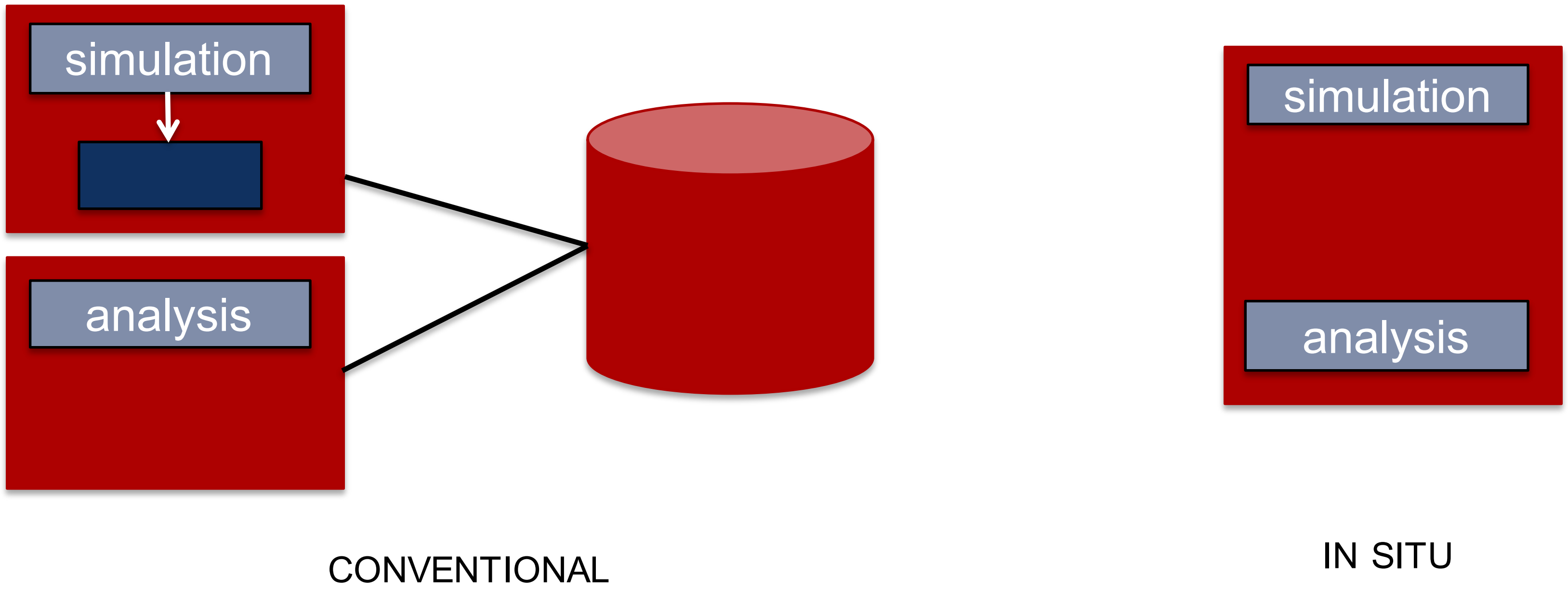


CONVENTIONAL

IN SITU

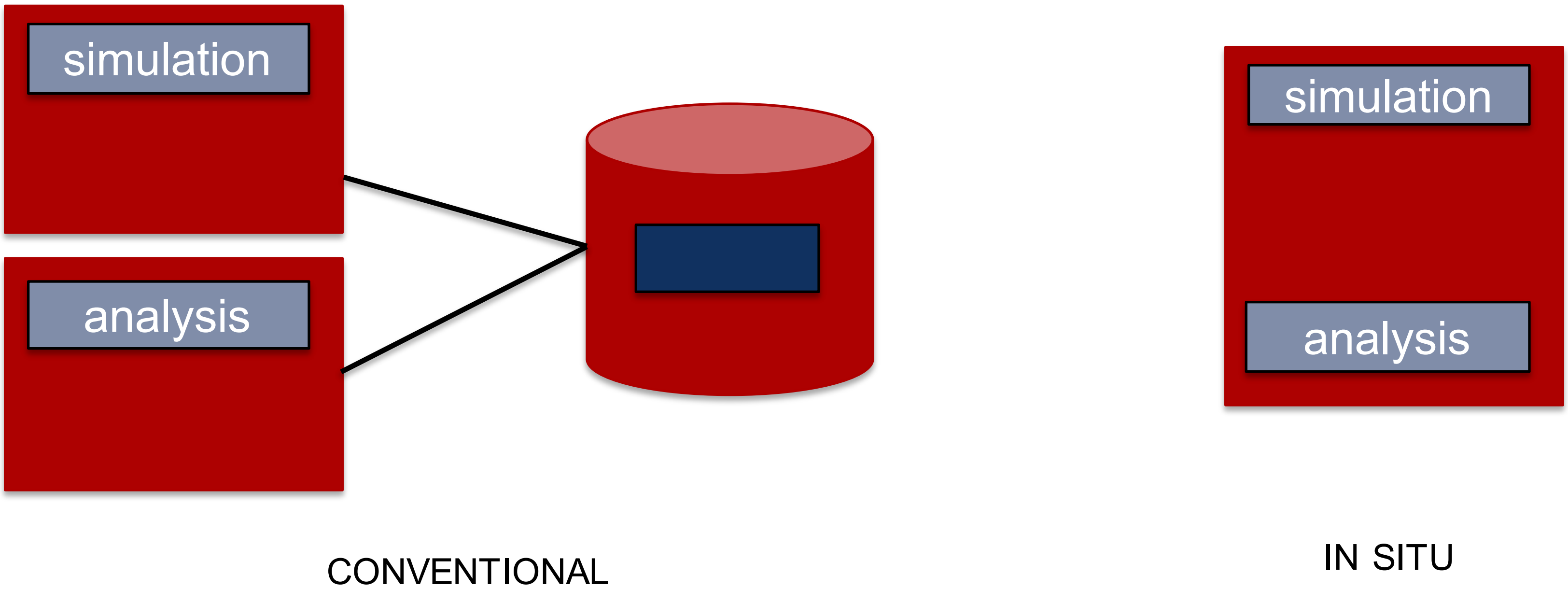
Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



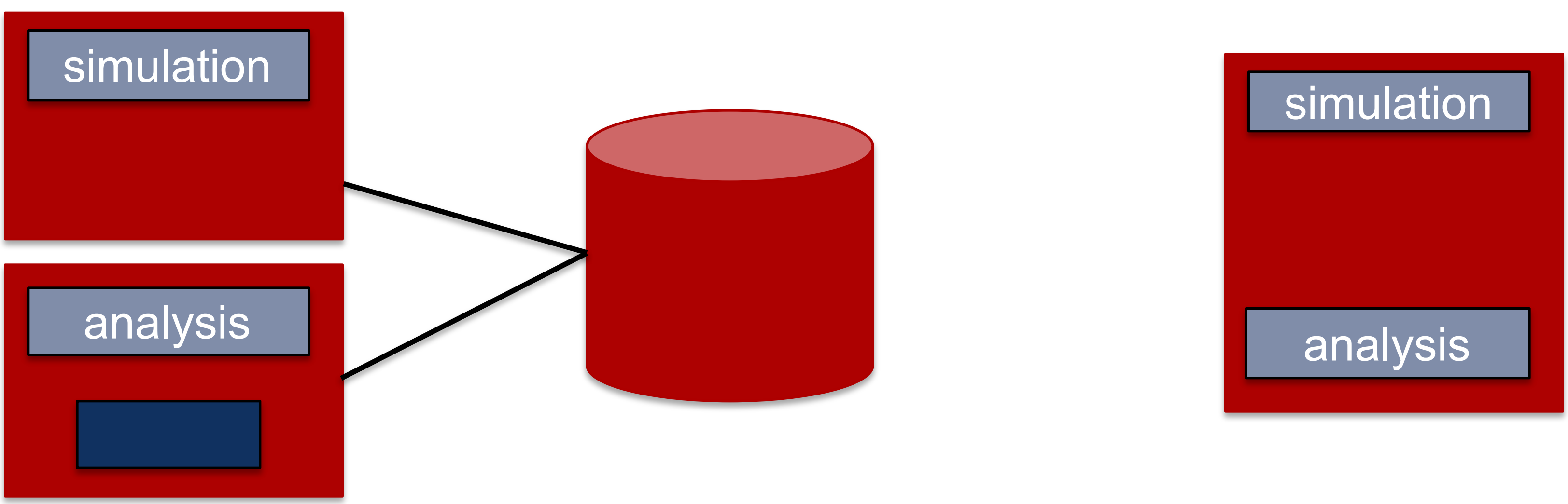
Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.

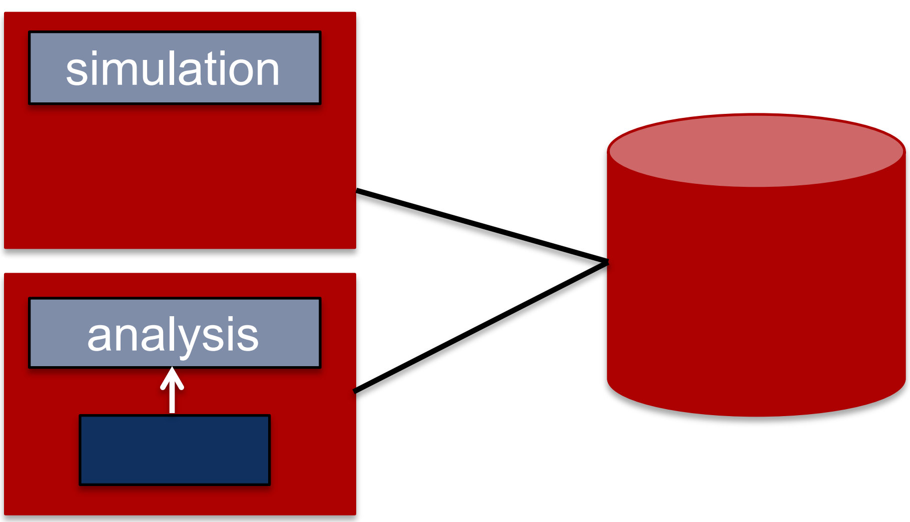


CONVENTIONAL

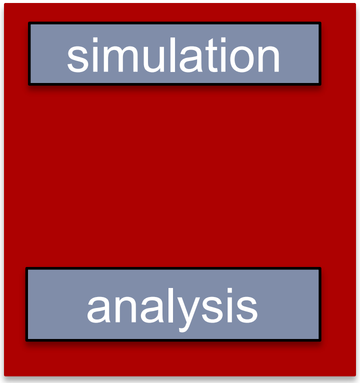
IN SITU

Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



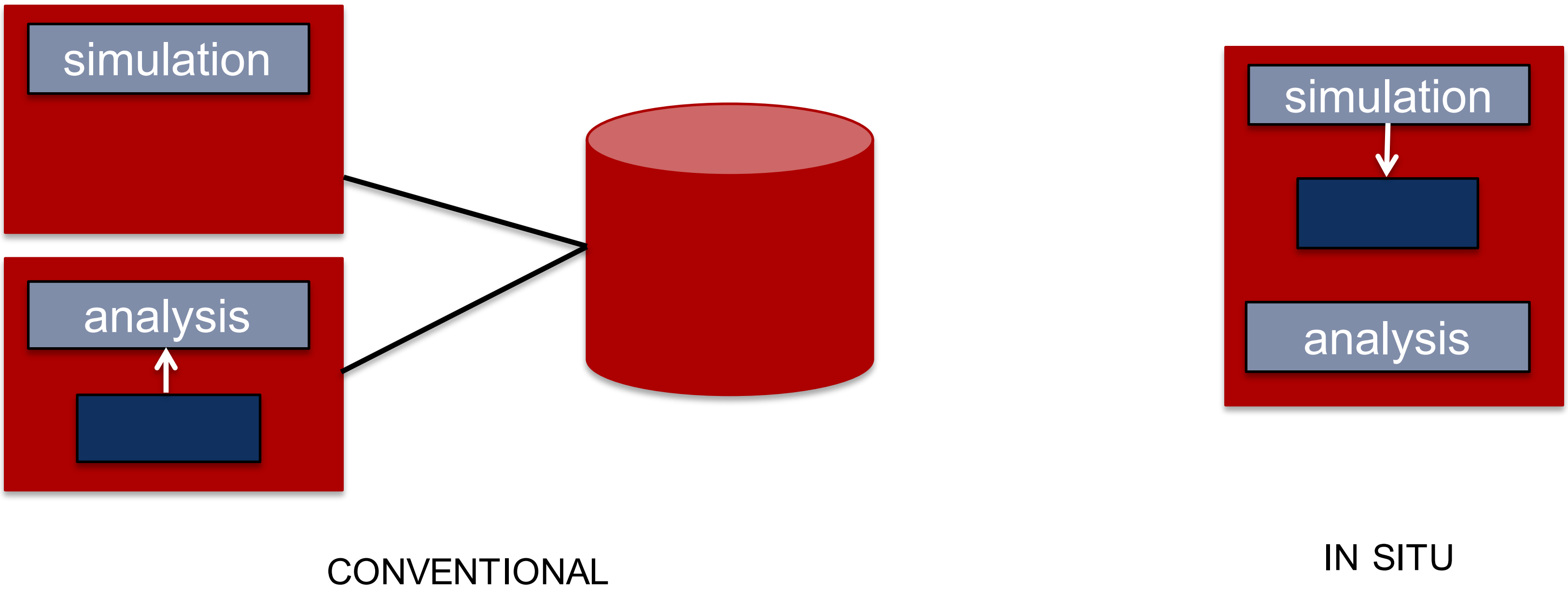
CONVENTIONAL



IN SITU

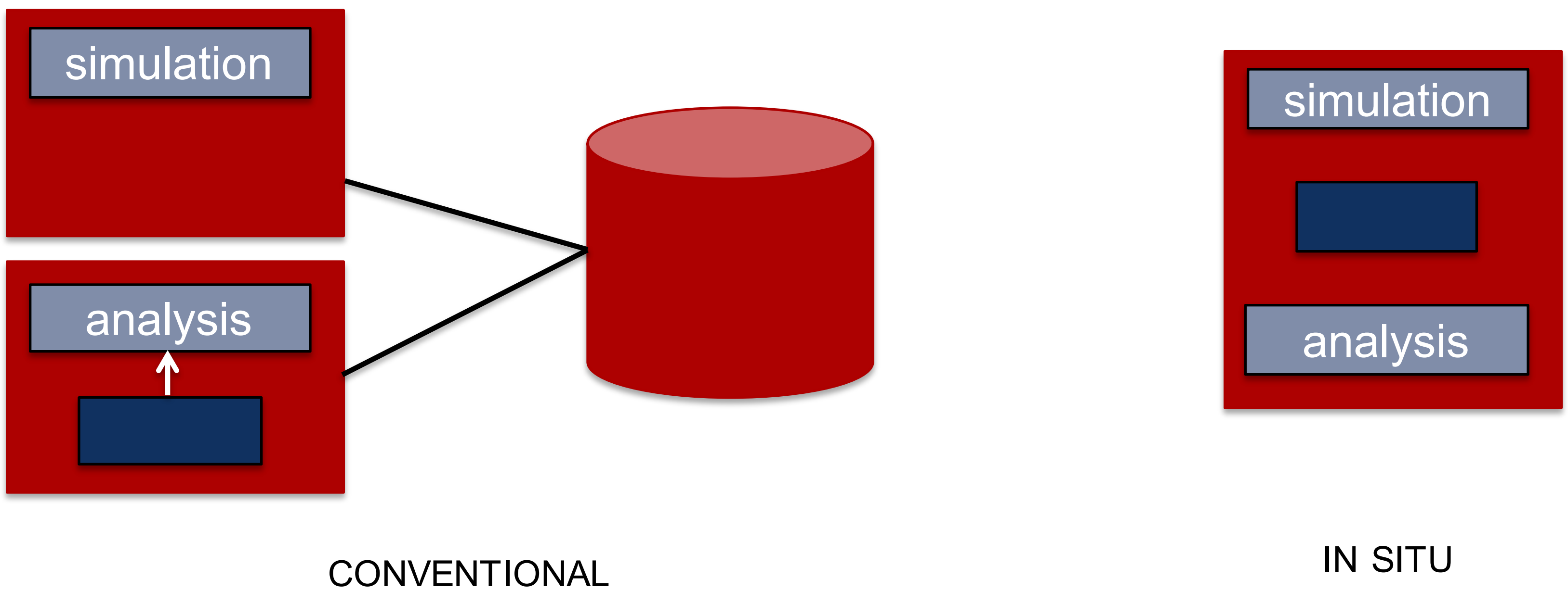
Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



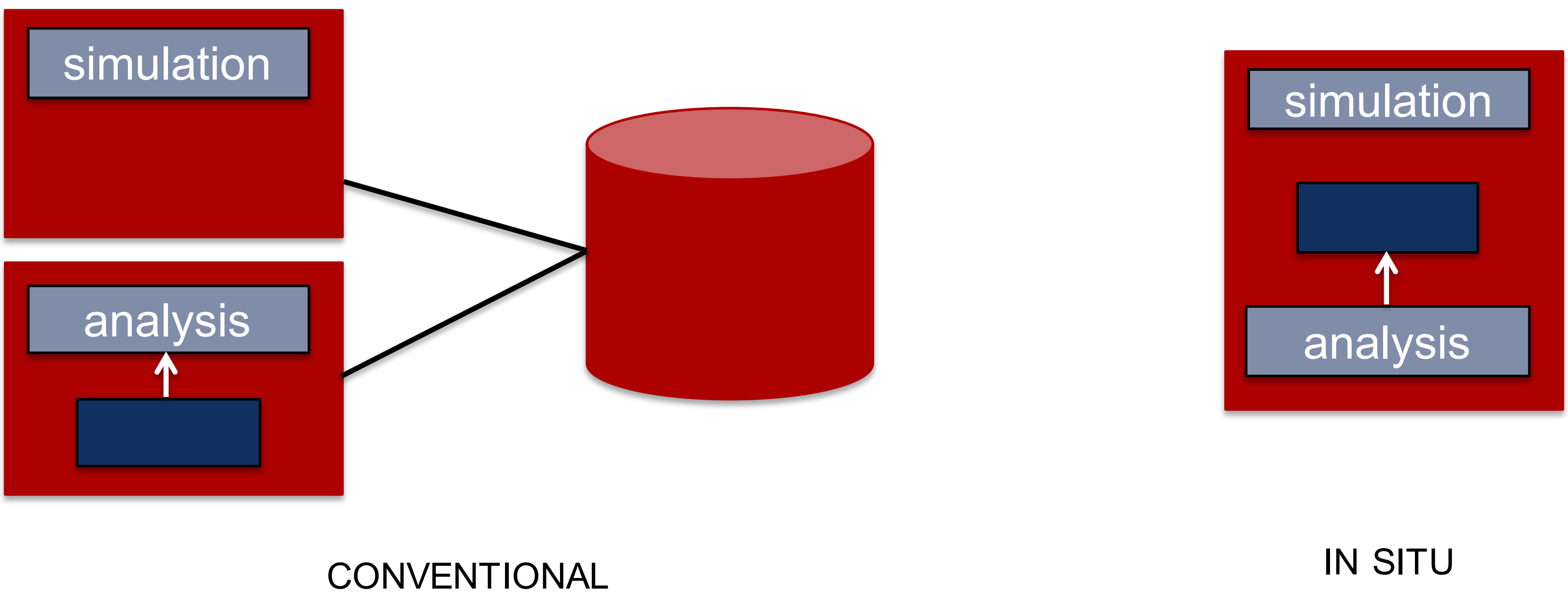
Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



Why In Situ Analytics?

- Currently, simulation codes commonly write output data to shared filesystem. Analysis reads from shared filesystem
- Data movement is expensive (limited I/O bandwidth, energy costs). Co-locating analysis with simulation eliminates unnecessary data movement.



Examples of In Situ Workloads

- Visualization
  - Selecting features of the output data that are necessary to generate images of simulation for human analysis
- Cosmology
  - Using parallel Voronoi tesellation to identify clusters and voids in the output of N-body simulations
- PreDatA
  - Middleware supporting the deployment of user-specified data processing (e.g., generating histograms)
- SmartPointer (Bonds)
  - Analysis of output generated by molecular dynamics codes. Bonds uses atom bonding information to identify and track cracks.

Examples of In Situ Workloads

- Visualization
  - Selecting features of the output data that are necessary to generate images of simulation for human analysis
- Cosmology
  - Using parallel Voronoi tesellation to identify clusters and voids in the output of N-body simulations
- PreDatA
  - Middleware supporting the deployment of user-specified data processing (e.g., generating histograms)
- SmartPointer (Bonds)
  - Analysis of output generated by molecular dynamics codes. Bonds uses atom bonding information to identify and track cracks.

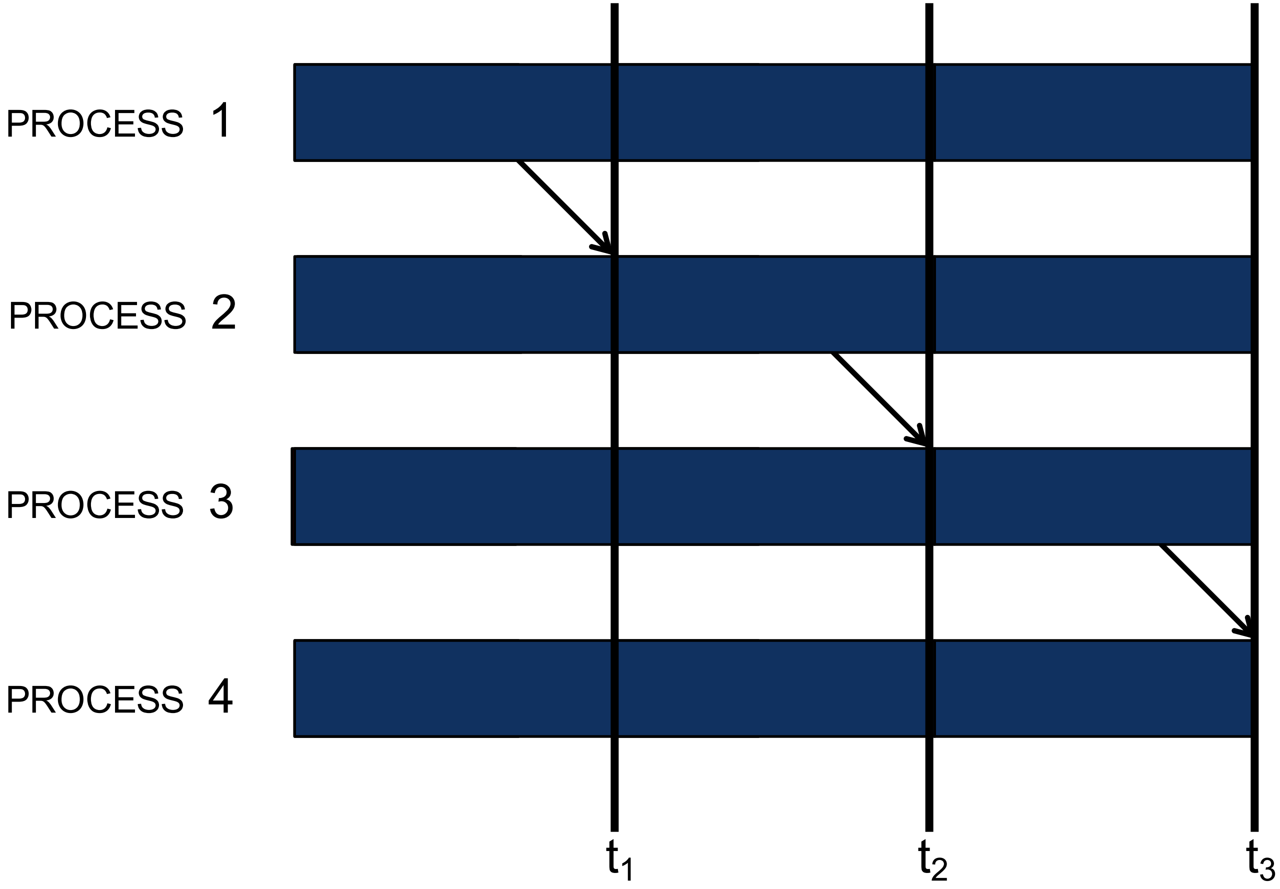
In Situ Analytics & Performance Interference

- Alternatives for co-locating analytics with simulation
  - TIME-SHARED : analytics and simulation running on same processor cores
  - SPACE-SHARED : subset of processors dedicated to analytics

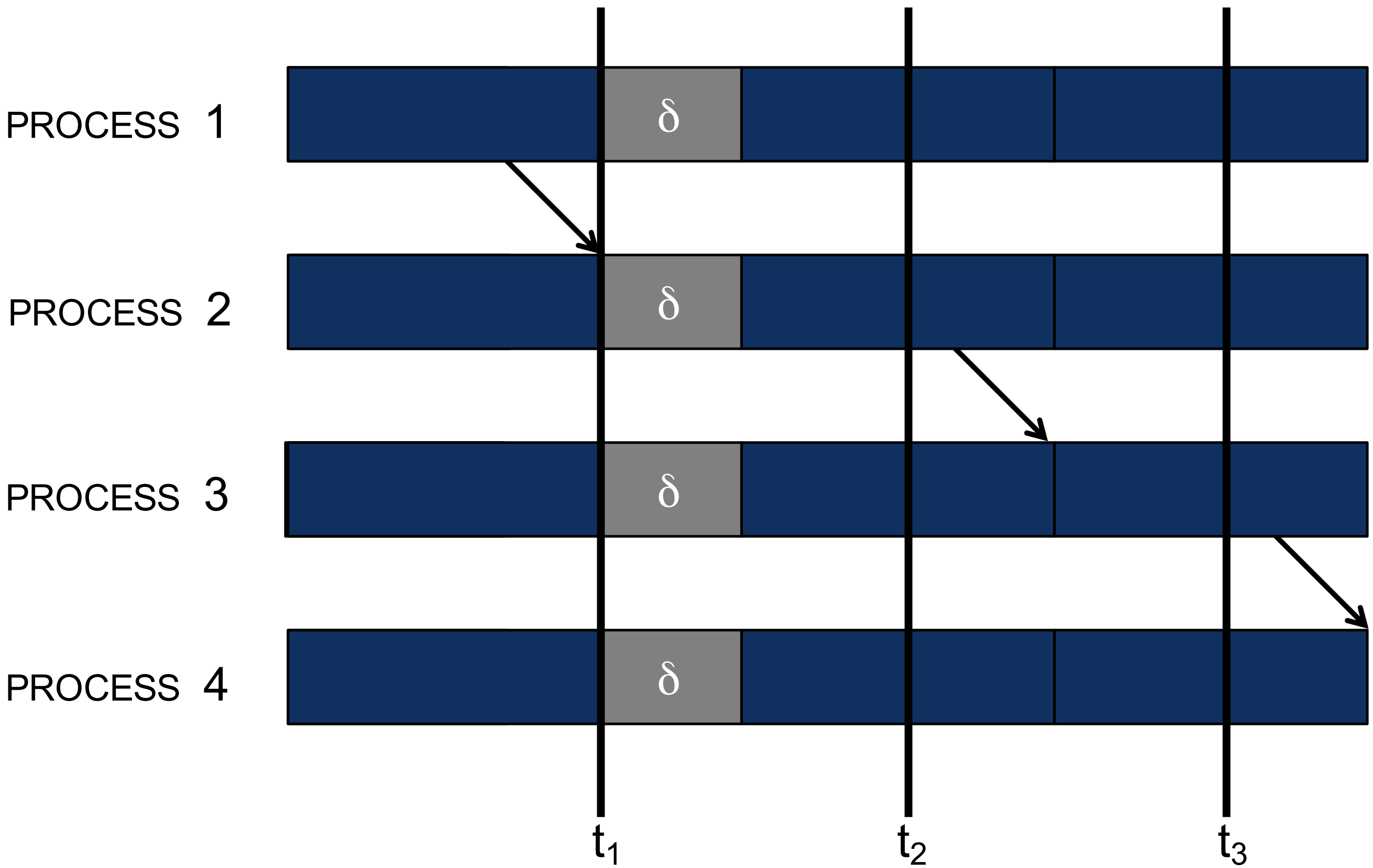
- In this paper, we examine time-shared in situ analytics; look for our work on space-shared analytics in the future

- Interrupting the simulation to run analysis may have disastrous performance consequences (cf. *OS noise*: Hoefler et al., SC10; Ferreira et al., SC08)

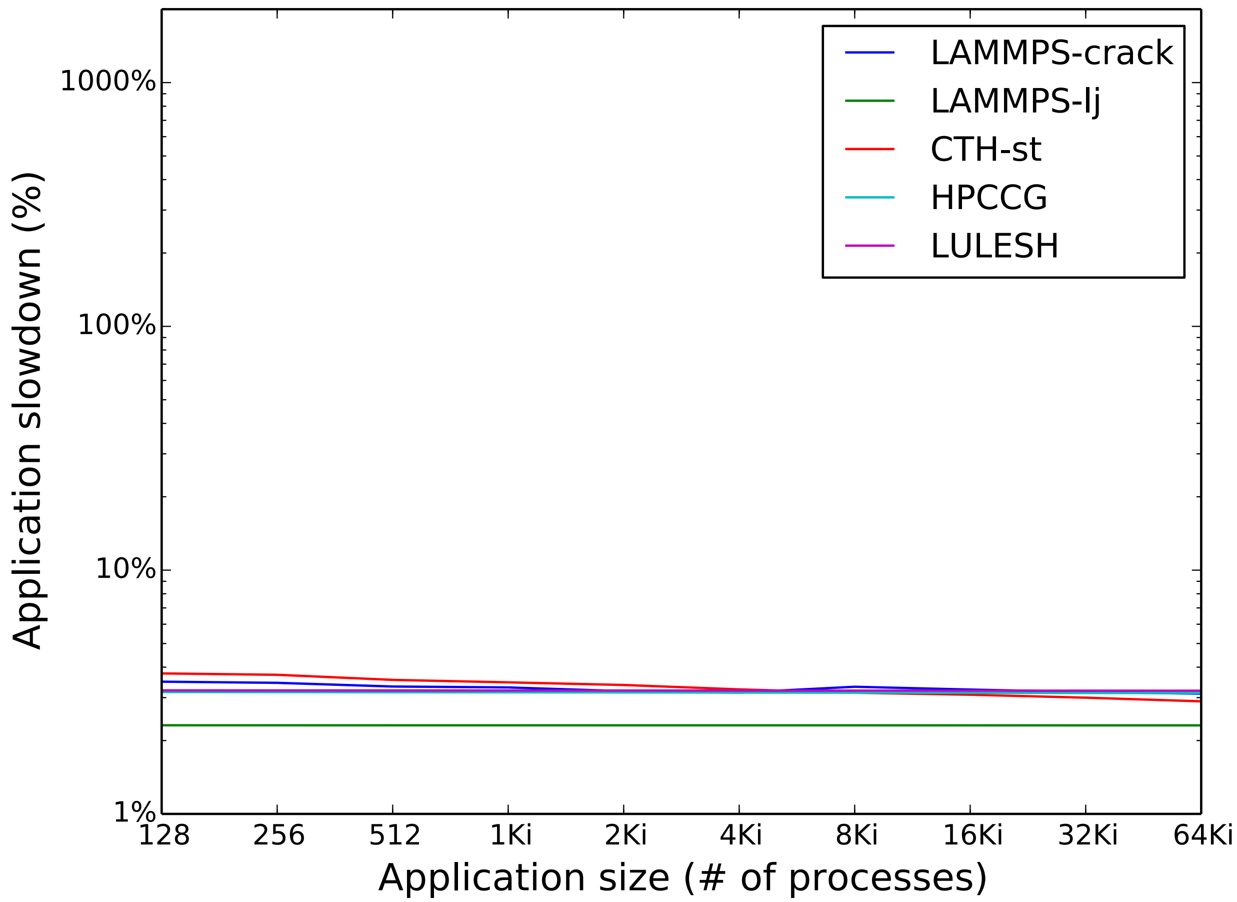
Perfectly Synchronous In Situ Analytics



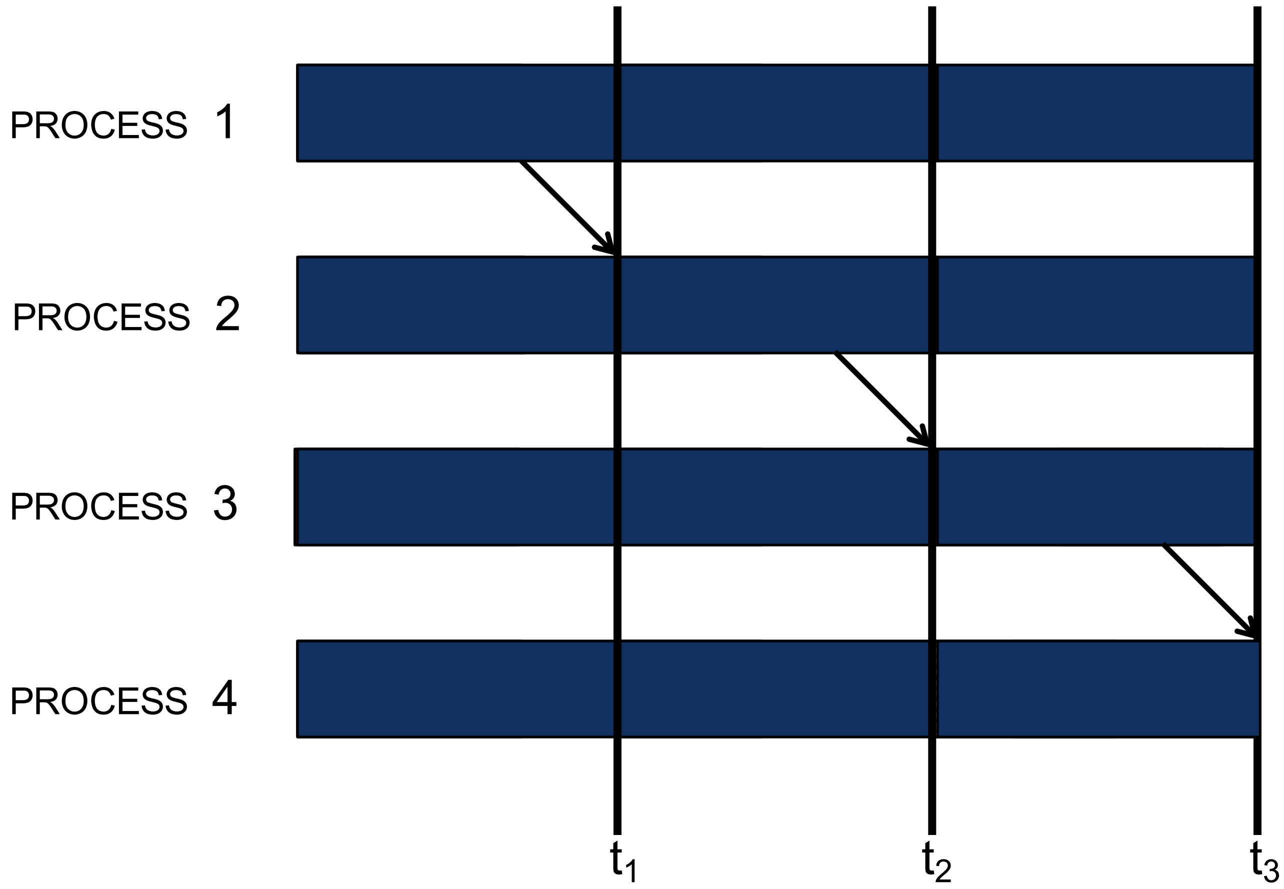
Perfectly Synchronous In Situ Analytics



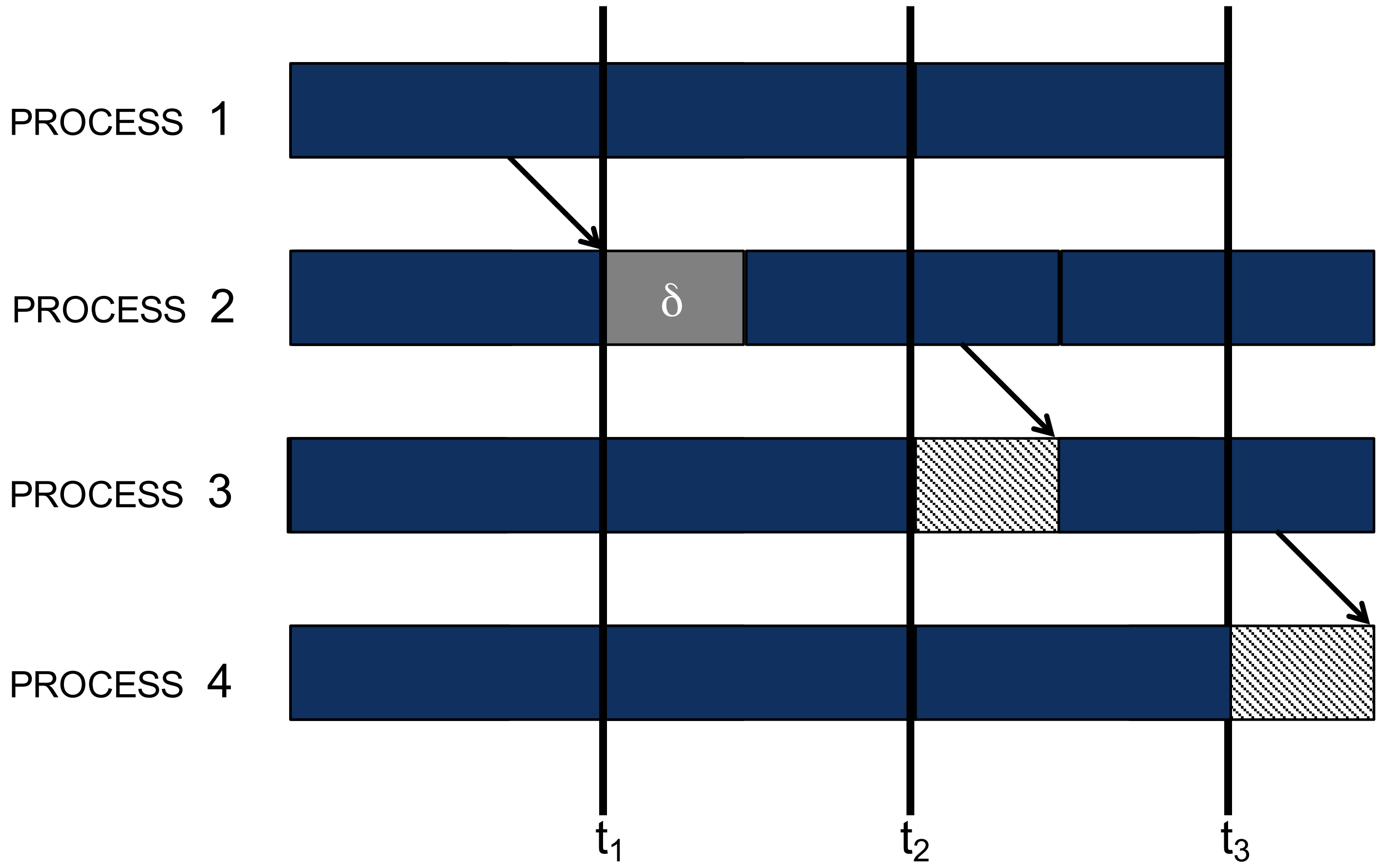
Perfectly Synchronous In Situ Analytics (cont'd)



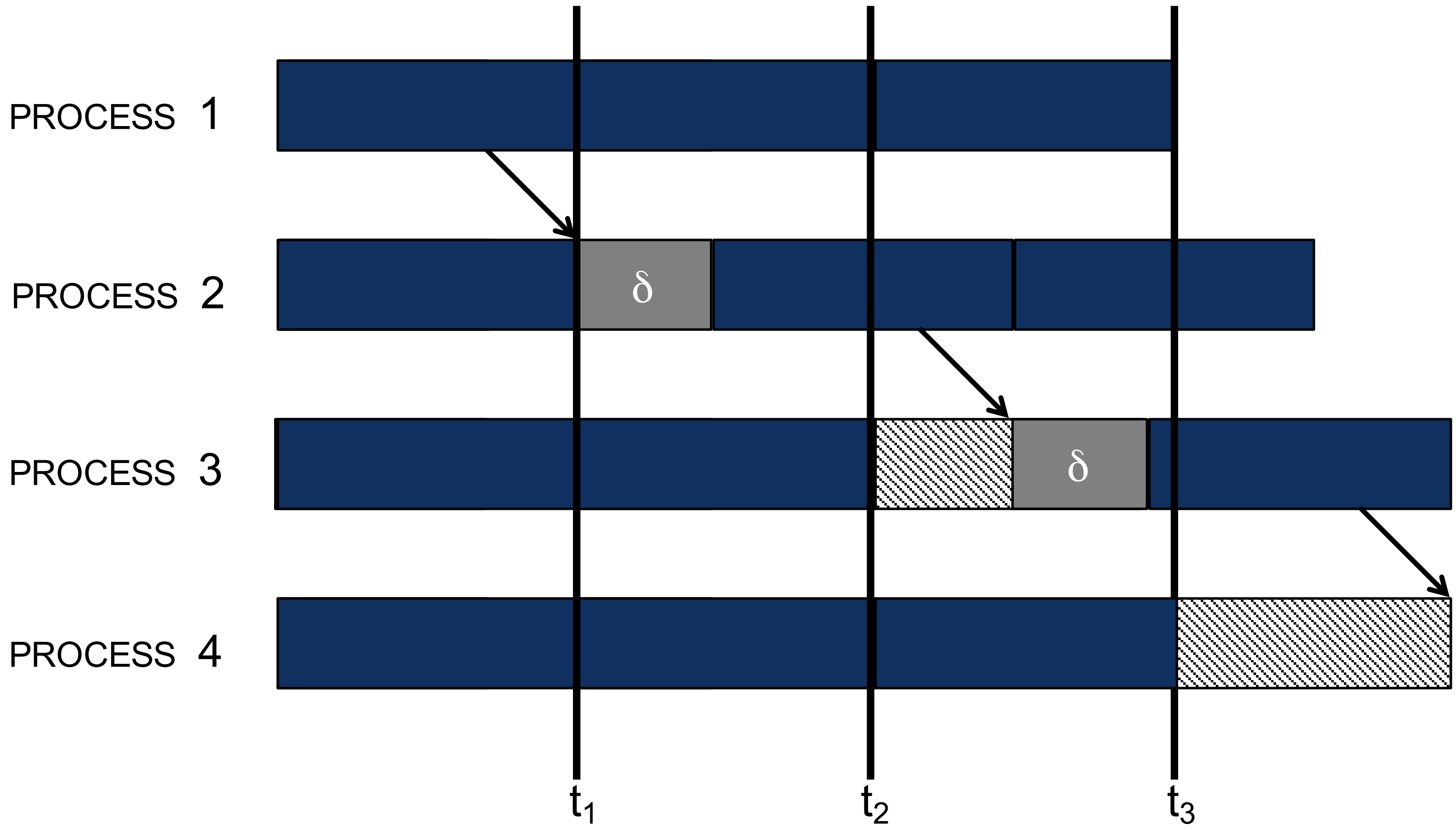
Completely Asynchronous In Situ Analytics



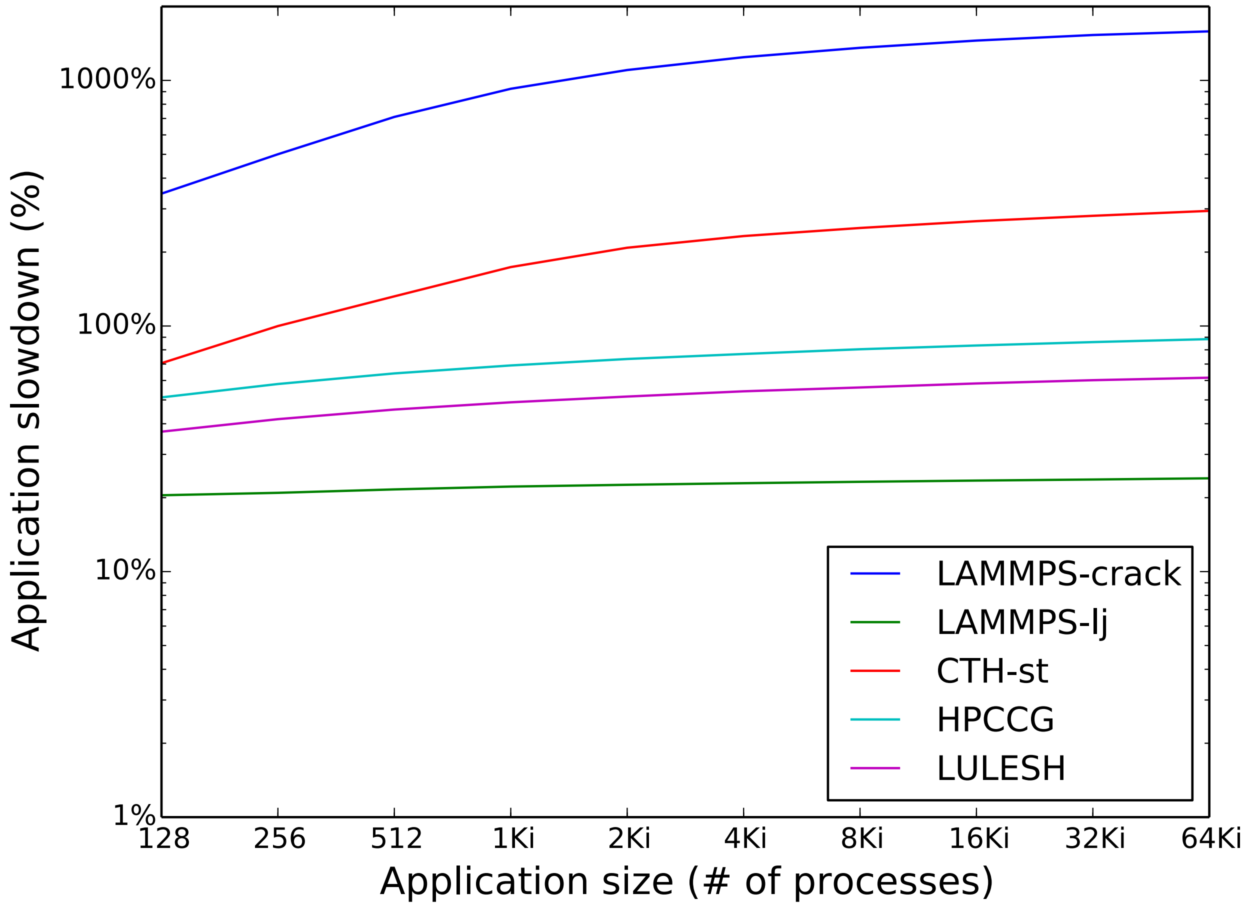
Completely Asynchronous In Situ Analytics



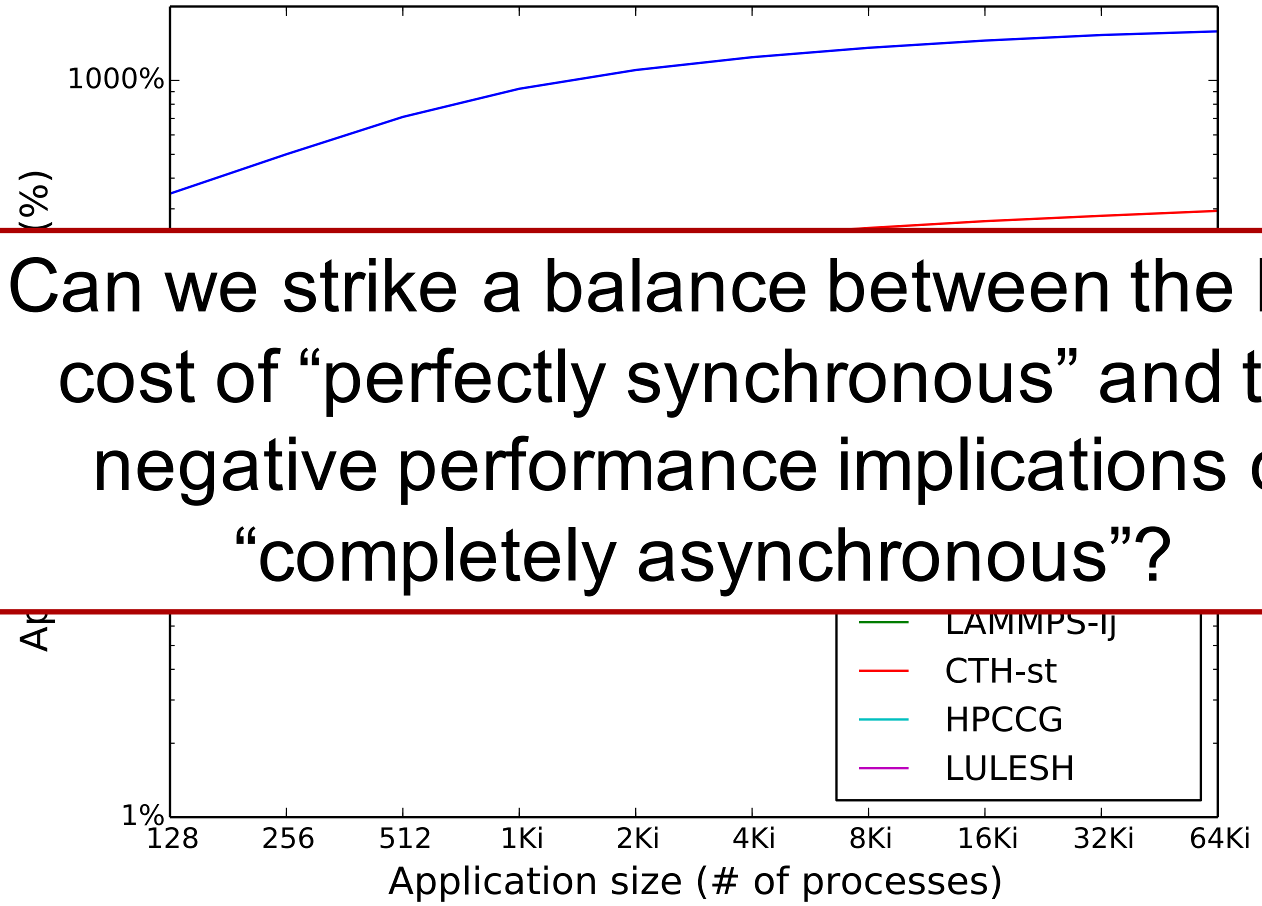
Completely Asynchronous In Situ Analytics



Completely Asynchronous In Situ Analytics (cont'd)



Completely Asynchronous In Situ Analytics (cont'd)



Can we strike a balance between the high cost of "perfectly synchronous" and the negative performance implications of "completely asynchronous"?

Collectives: Algorithms vs. Operations

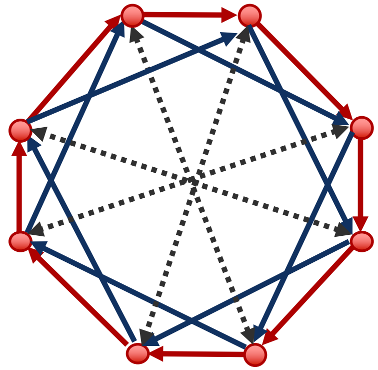
- MPI 3.0 section 5.1

  It is dangerous to rely on synchronization side-effects of the collective operations for program correctness. … On the other hand, a correct, portable program must allow for the fact that a collective call may be synchronizing. Though one cannot rely on any synchronization side-effect, one must program so as to allow it.

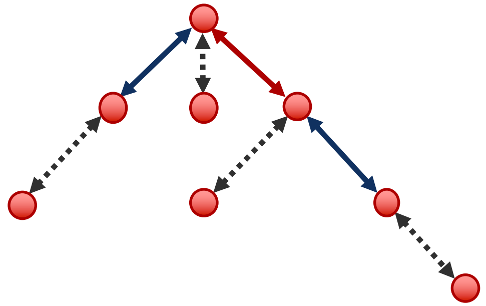
- Therefore, we explicitly analyze the synchronizing effects of collective *algorithms* rather than collective *operations*

Collective Algorithms

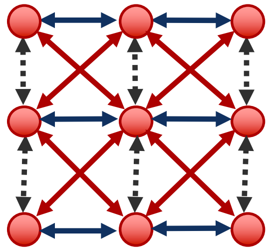
- Dissemination
  (e.g., to implement `MPI_Allreduce`)



- Binomial tree dispersal/aggregation
  (e.g., to implement `MPI_Bcast`/`MPI_Reduce`)



- Stencil communication
  (e.g., to implement `MPI_Neighbor_alltoall`)

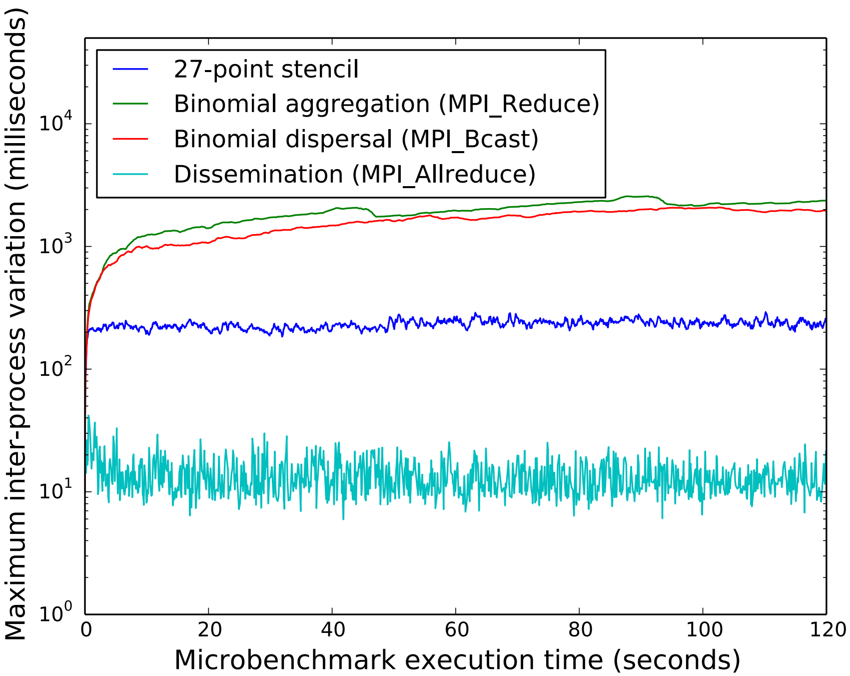


Experimental Approach

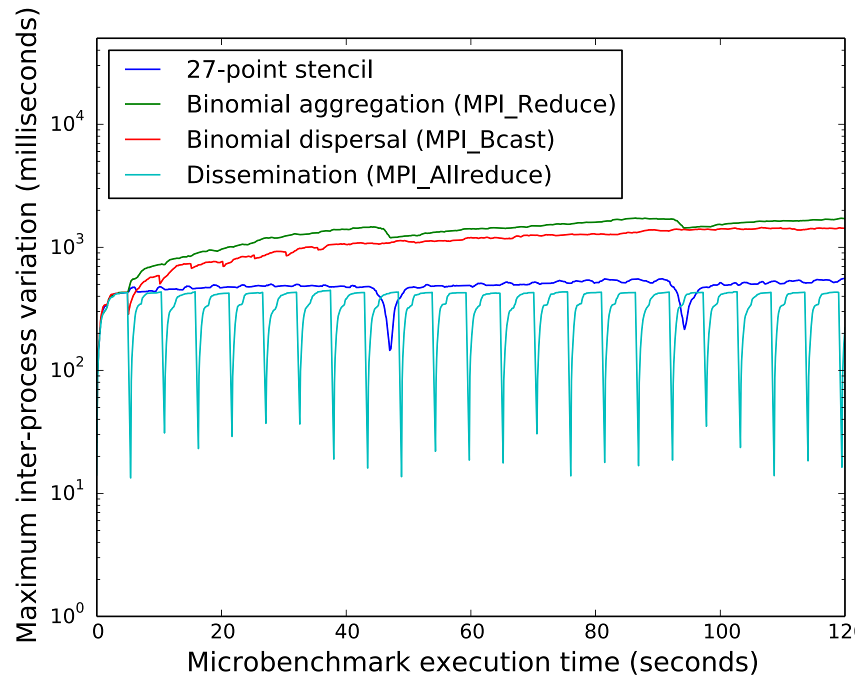
- Simulate application execution using LogGOPSim (Hoefler et al., LSAP 2010; *see also* Levy et al., PMBS 2013)
- Examine five workloads
  - LAMMPS
    - Molecular dynamics simulation from Sandia National Laboratories. We used the LAMMPS 2D crack and Lennard-Jones (LJ) potentials.
  - CTH
    - Application from Sandia National Laboratories for modeling complex problems that are characterized by large deformations or strong shocks
  - HPCCG
    - Conjugate gradient solver from the Mantevo suite of mini-applications
  - LULESH
    - An application that represents the behavior of a typical hydrocode

Collective Algorithm-induced Synchronization

- Microbenchmark that allows us to vary collective frequency
- Dissemination has the greatest synchronizing effect
- More frequent collectives generally result in tighter synchronization



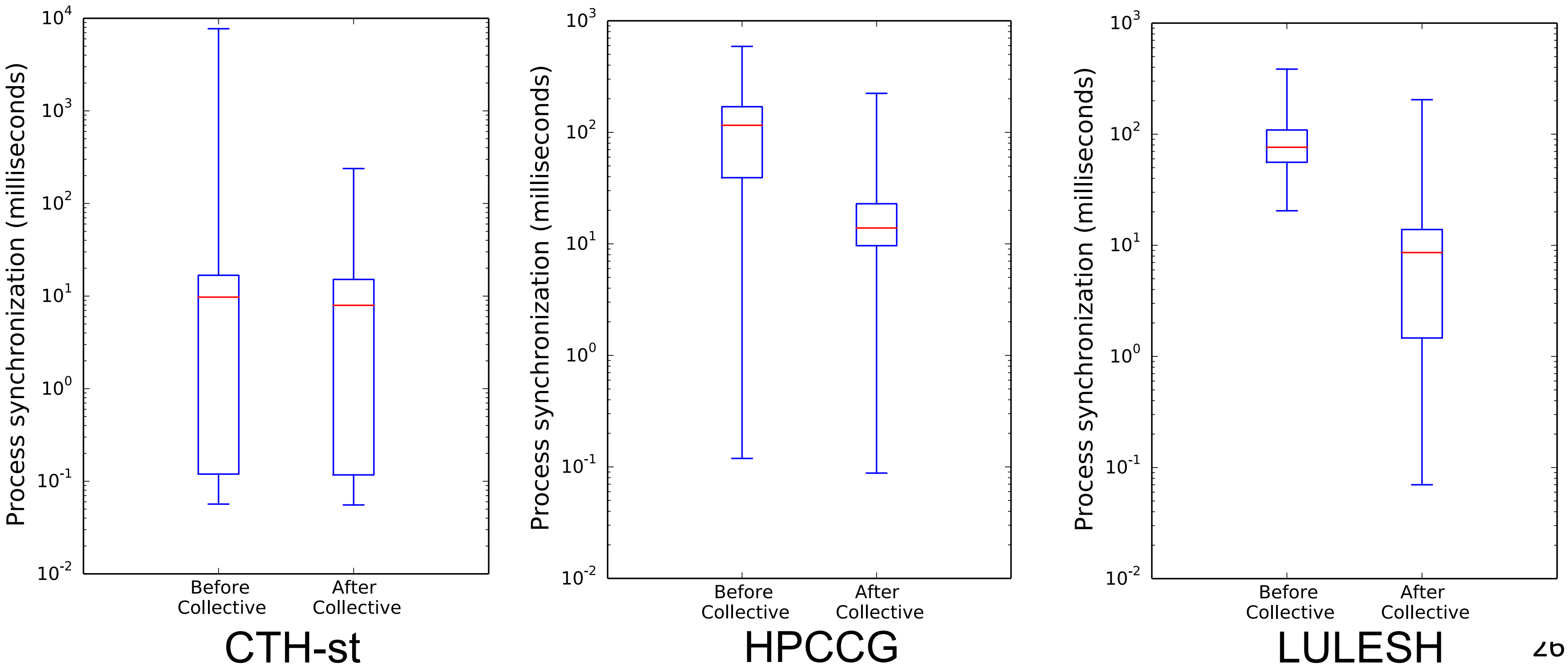
50 millisecond collective period



5 second collective period

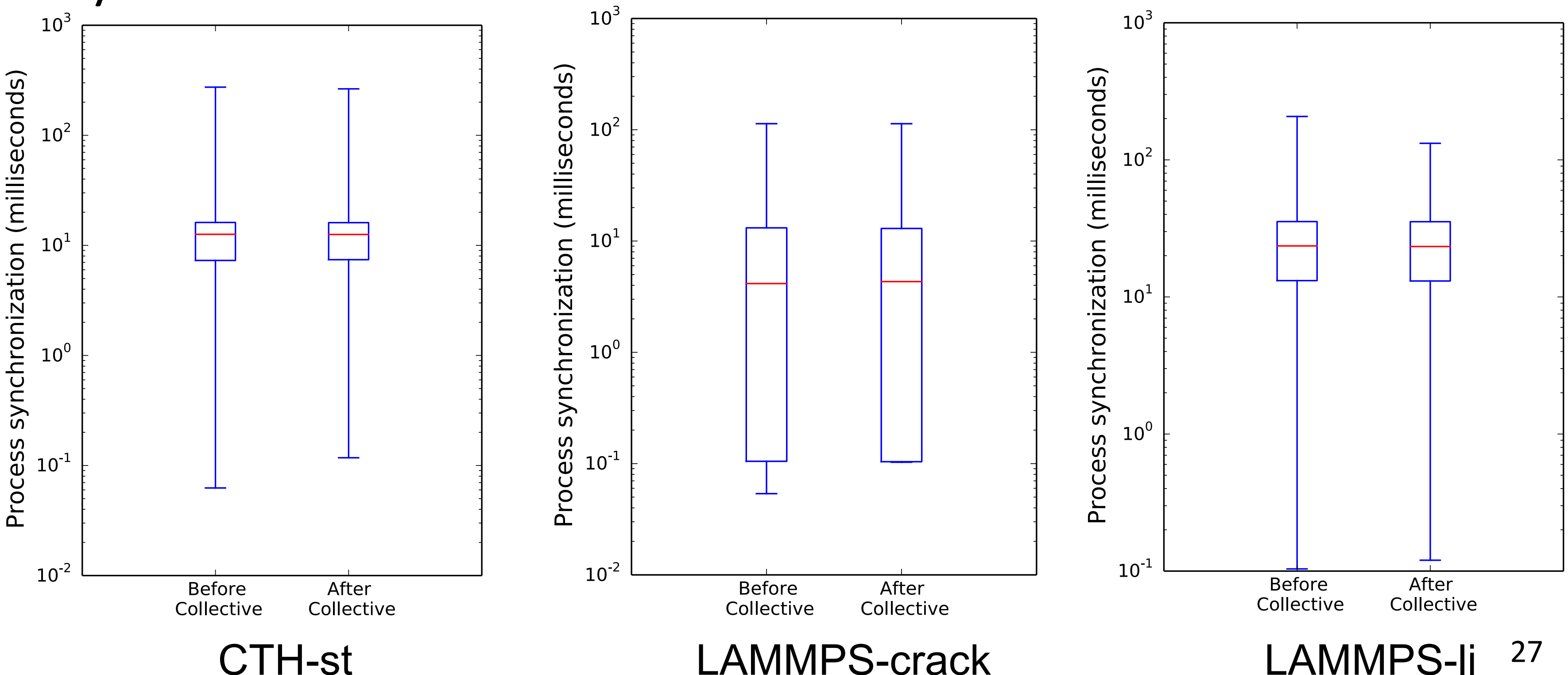
Application-level Synchronization (Dissemination)

- Used simulation to measure the impact of dissemination algorithm on process synchronization
- In most cases, dissemination synchronizes processes to within 10s of milliseconds



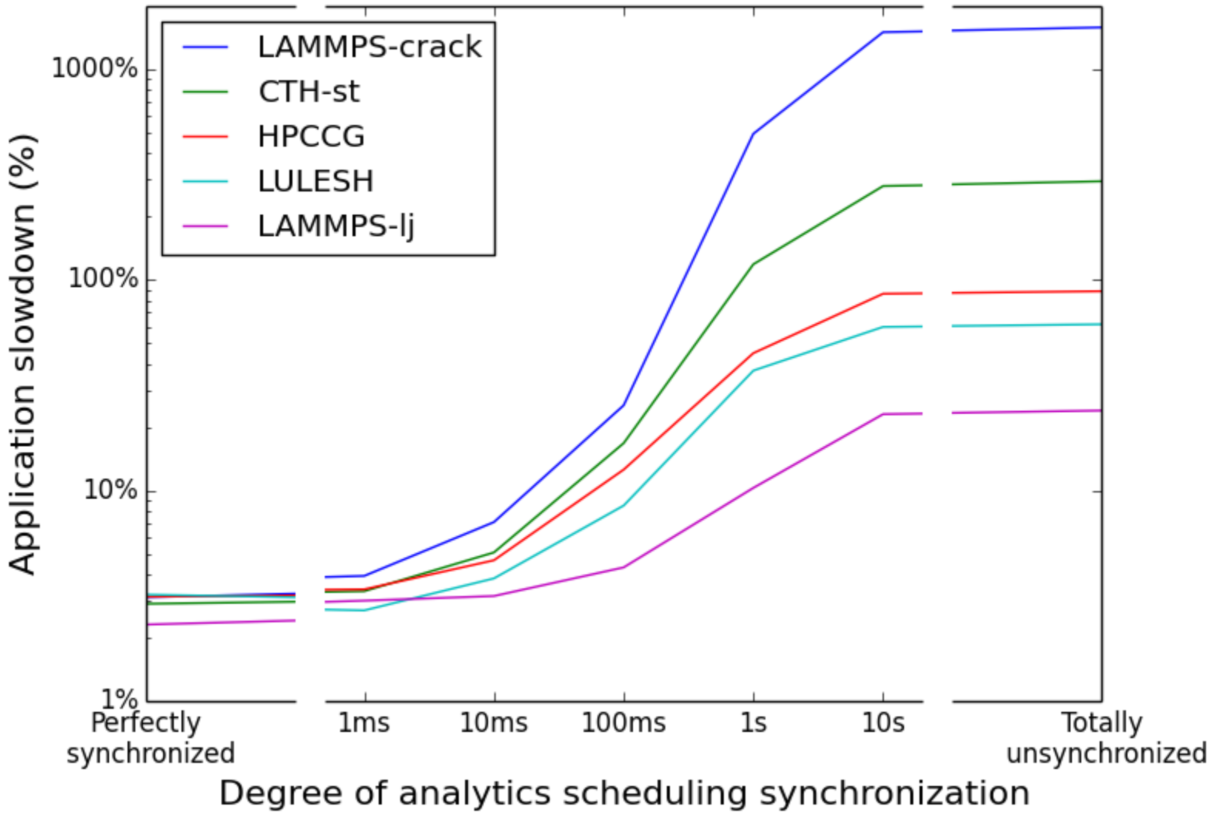
Application-level Synchronization (Binomial dispersal)

- Used simulation to measure impact of binomial dispersal algorithm on process synchronization
- Binomial dispersal has little impact on process synchronization



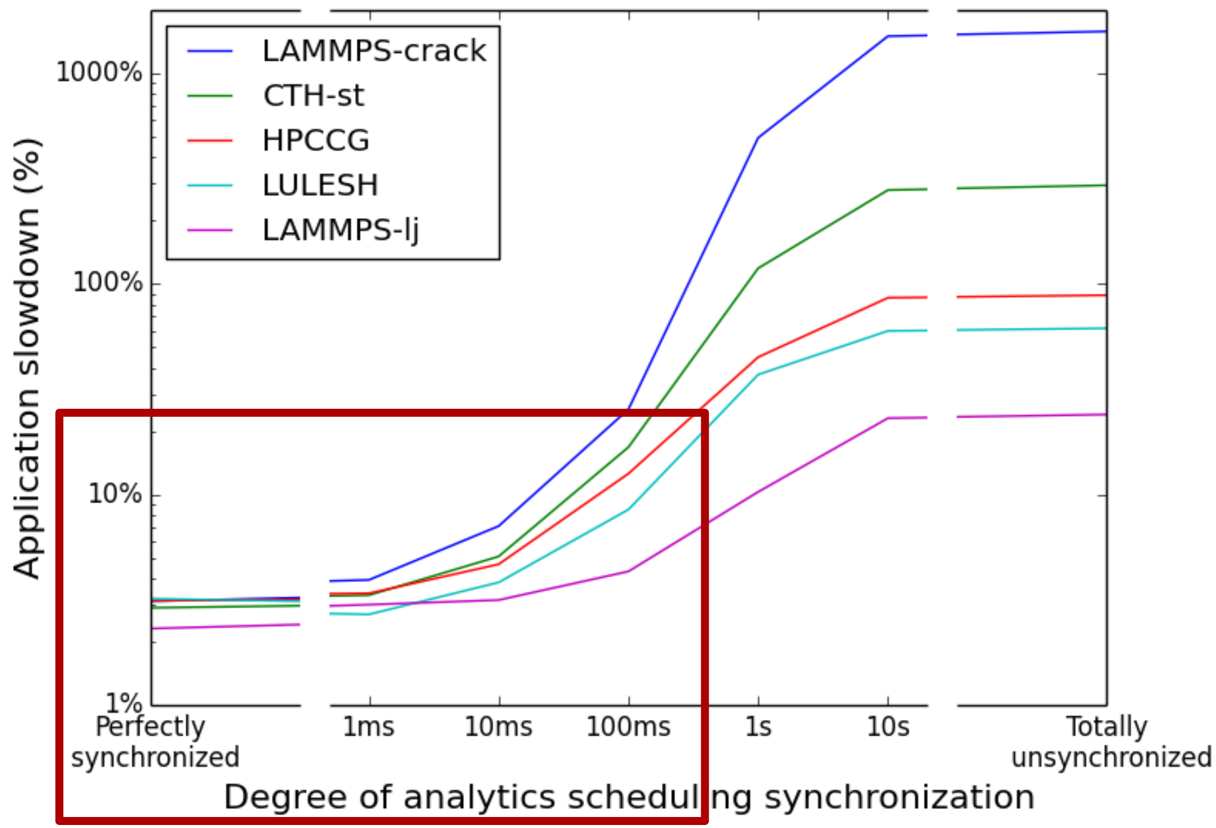
Synchronizing Analytics

- Even modest synchronization can significantly reduce the impact of executing analytics



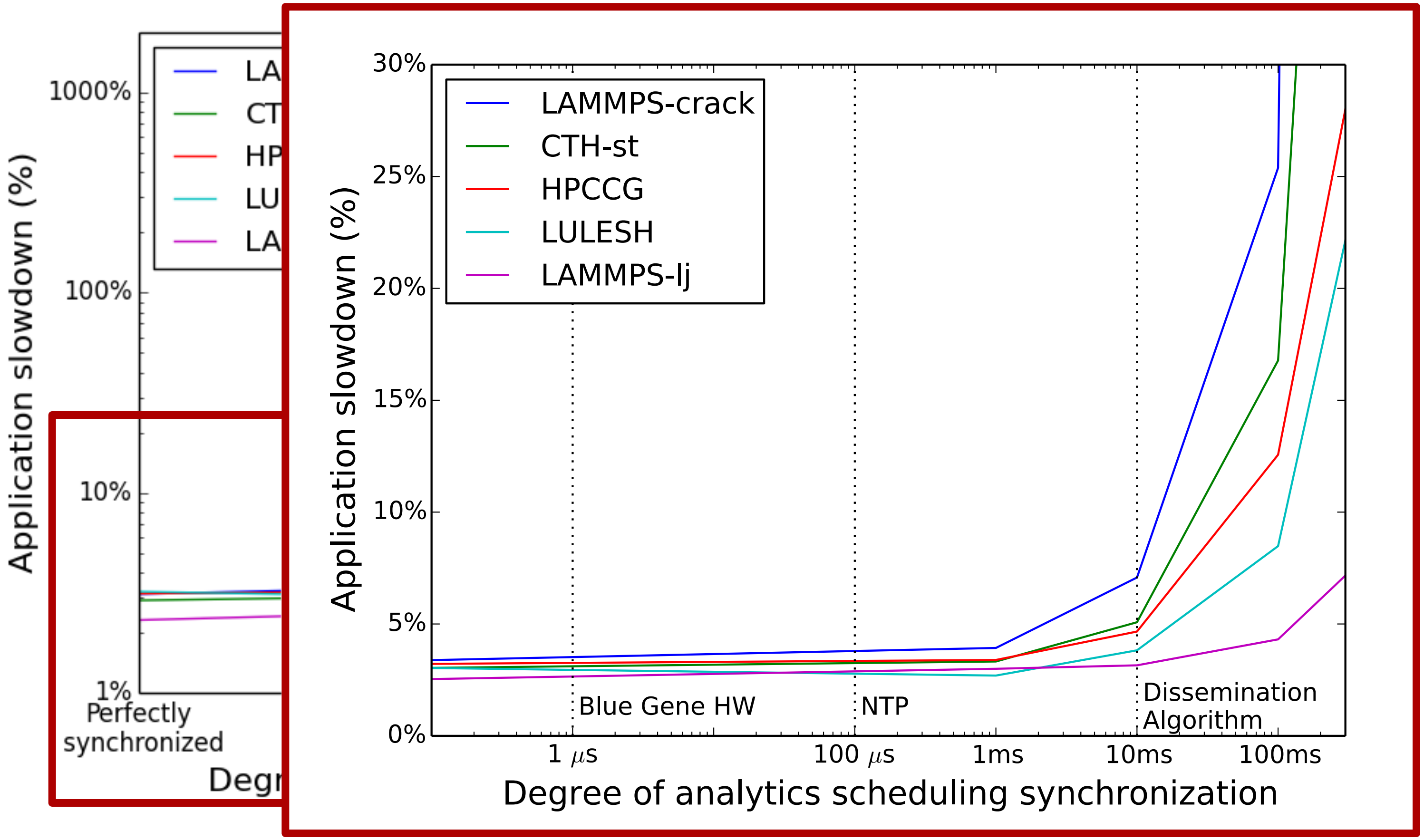
Synchronizing Analytics

- Even modest synchronization can significantly reduce the impact of executing analytics



Synchronizing Analytics

- Even modest synchronization can significantly reduce the impact of executing analytics



Conclusion

- Perfectly synchronizing the execution of time-shared analytics tasks minimizes impact, but may be expensive to achieve; executing analytics tasks with no synchronization can have disastrous performance impacts.
- Some collective algorithms (e.g., dissemination, high-dimension stencils) have the effect of approximately synchronizing application execution; others (e.g., binomial dispersal/aggregation) have little effect on process synchronization
- Even modest synchronization (e.g., within 10s of milliseconds) can dramatically reduce the performance degradation caused by time-shared analytics; expensive synchronization methods are unnecessary

Co-authors

- Kurt B. Ferreira
  Sandia National Laboratories

- Patrick Widener
  Sandia National Laboratories

- Patrick G. Bridges
  University of New Mexico

- Oscar H. Mondragon
  University of New Mexico

Questions?

sllevy@sandia.gov
www.sandia.gov/~sllevy