SAND2014-19851C

Early Experiences Co-Scheduling Work and Communication Tasks for Hybrid MPI+X Applications

Dylan Stark, R. Barrett, R. Grant, S. Olivier, K. Pedretti, C. Vaughan
Sandia National Laboratories

ExaMPI Workshop

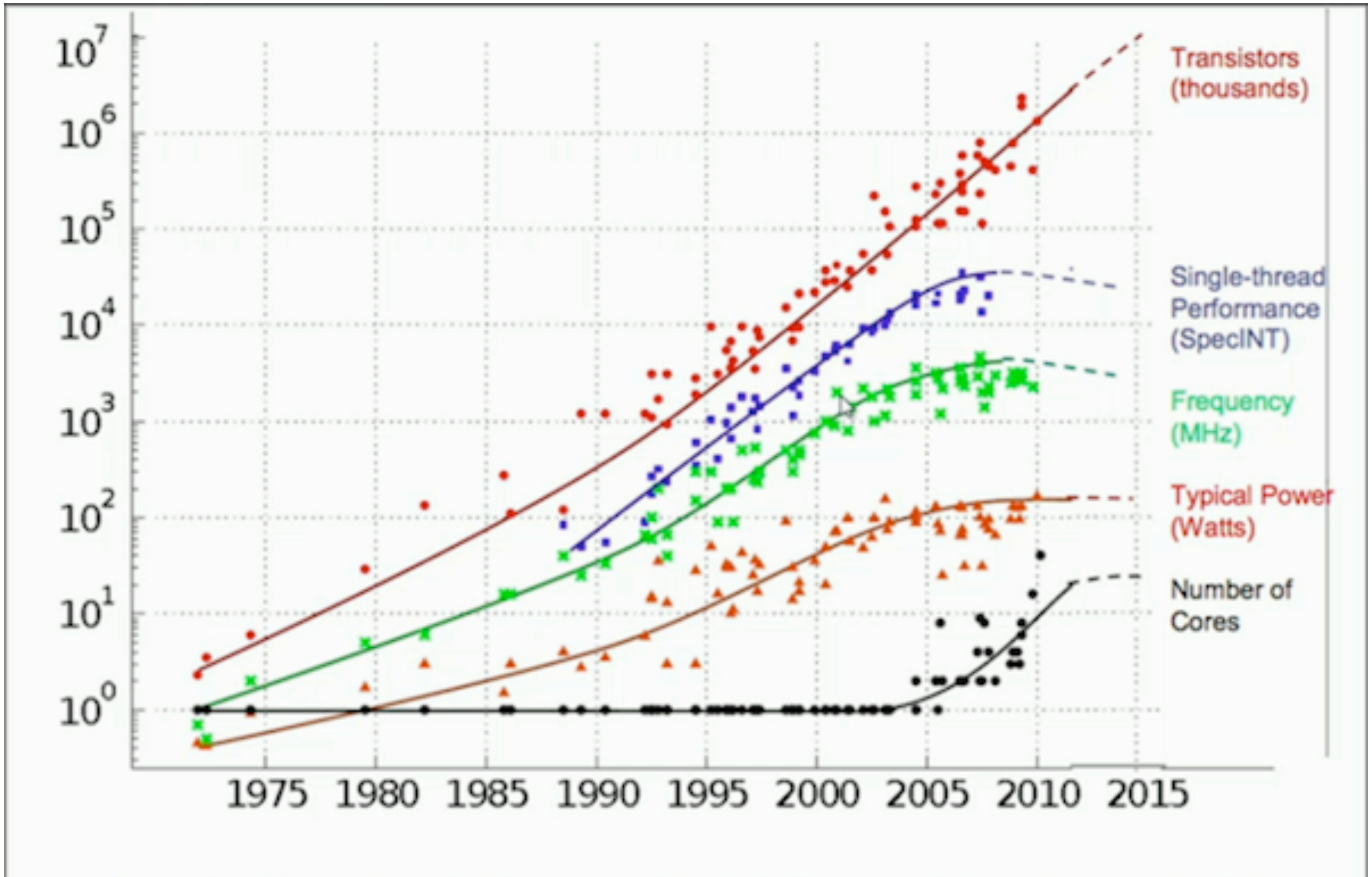
SC14

November 17, 2014

Sandia National Laboratories

Exceptional service in the national interest

End of an era …



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

… where to start?

Level	**Parallelism**	Coordination
SINGLE		
FUNNELED	Single call site	User’s problem
SERIALIZED	Multiple calls sites, app-level mutex	User’s problem
MULTIPLE	Multiple call sites, MPI-level mutex	MPI’s problem

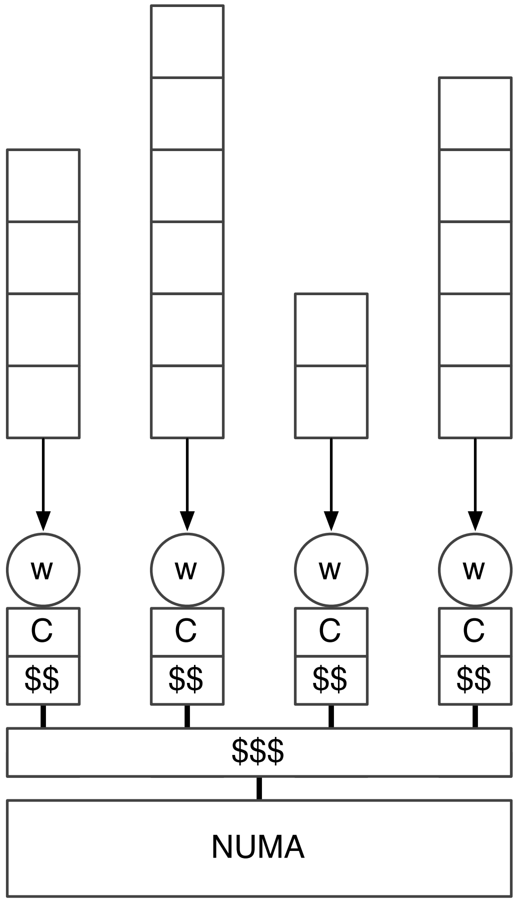
Leverage low-level threading runtime (LLTR)

Level	Parallelism	Coordination
SINGLE		
FUNNELED	Single call site	~~User's problem~~ **LLTR's problem**
SERIALIZED	Multiple calls sites, app-level mutex	~~User's problem~~ **LLTR's problem**
MULTIPLE	Multiple call sites, MPI-level mutex	~~MPI's problem~~ **LLTR's problem**

Specifically: cooperative user-level threading (tasking)

Challenges:

- State management
- Context swapping
- Memory allocation
- System call interception
- Scheduling
- Synchronization
- Locality
- Hardware affinity
- Concurrent data structures
- Adaptivity
- …
- Configurability
- Portability



Work stealing

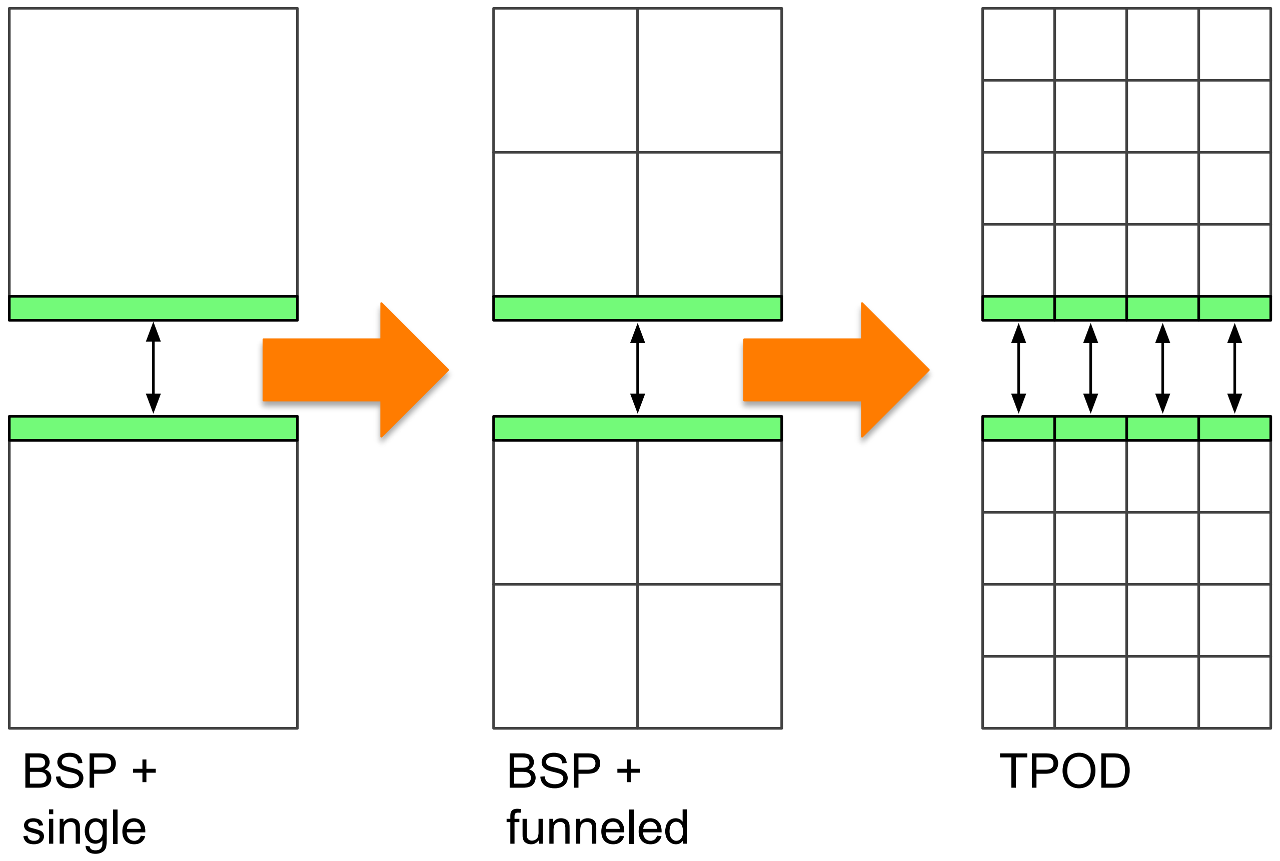
Work queues

Worker (Pthreads)

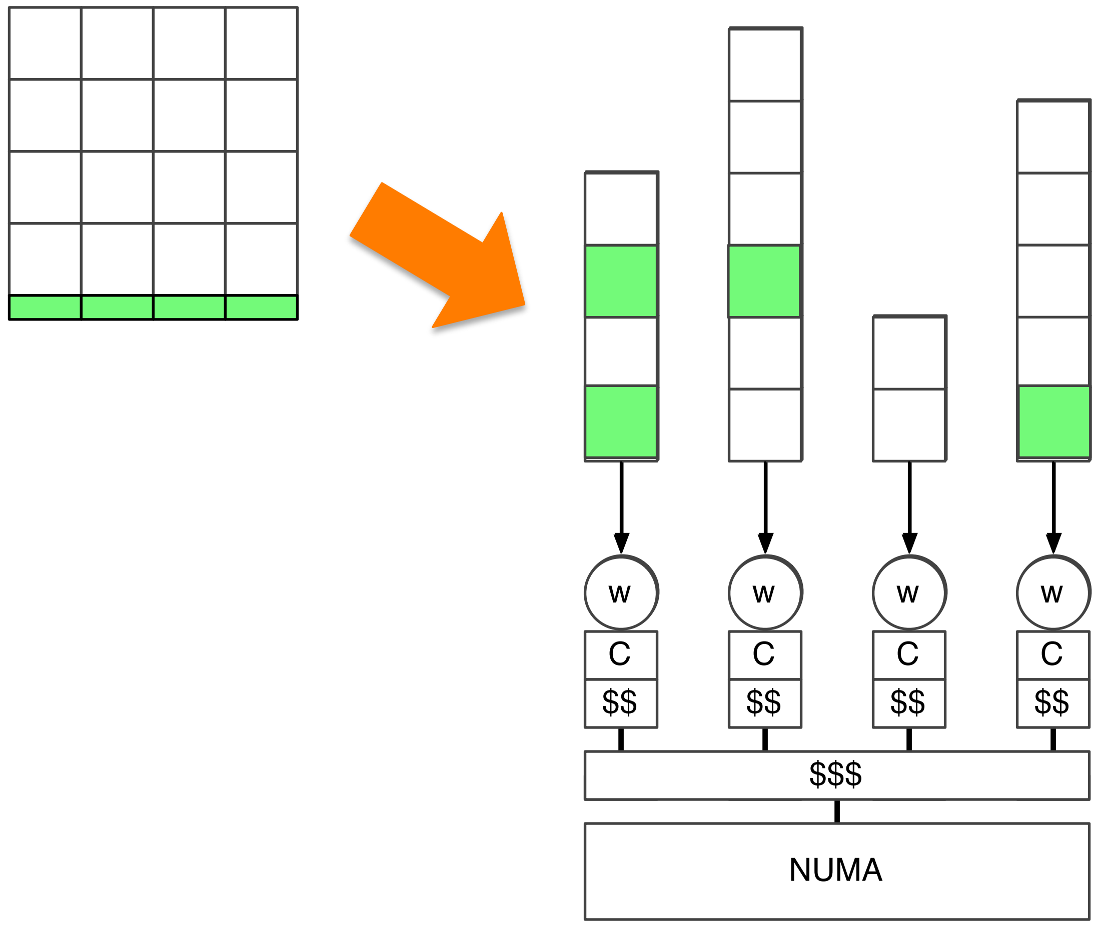
Evaluation using a proxy application

CTH
(shock hydro.)

miniGhost
(halo exchange)



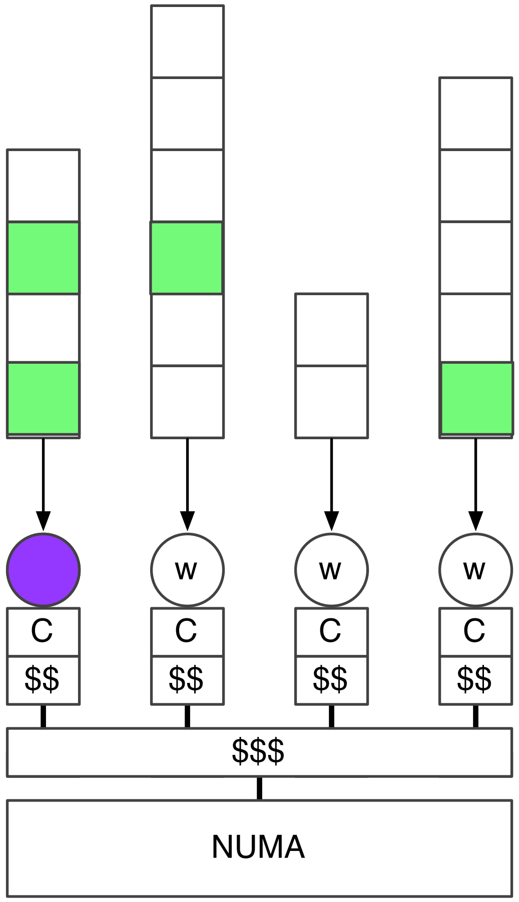
Design of LLTR



Design of LLTR

Challenges:

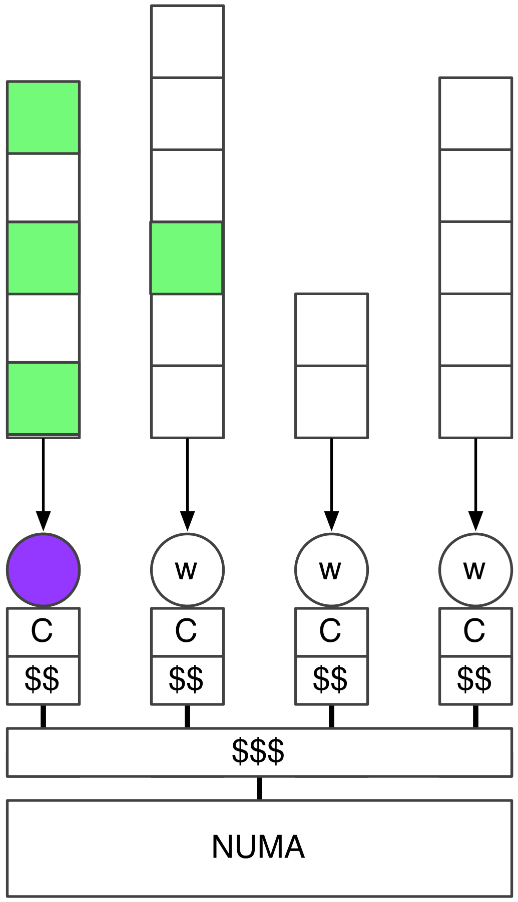
- Ensuring thread safety



Design of LLTR

Challenges:

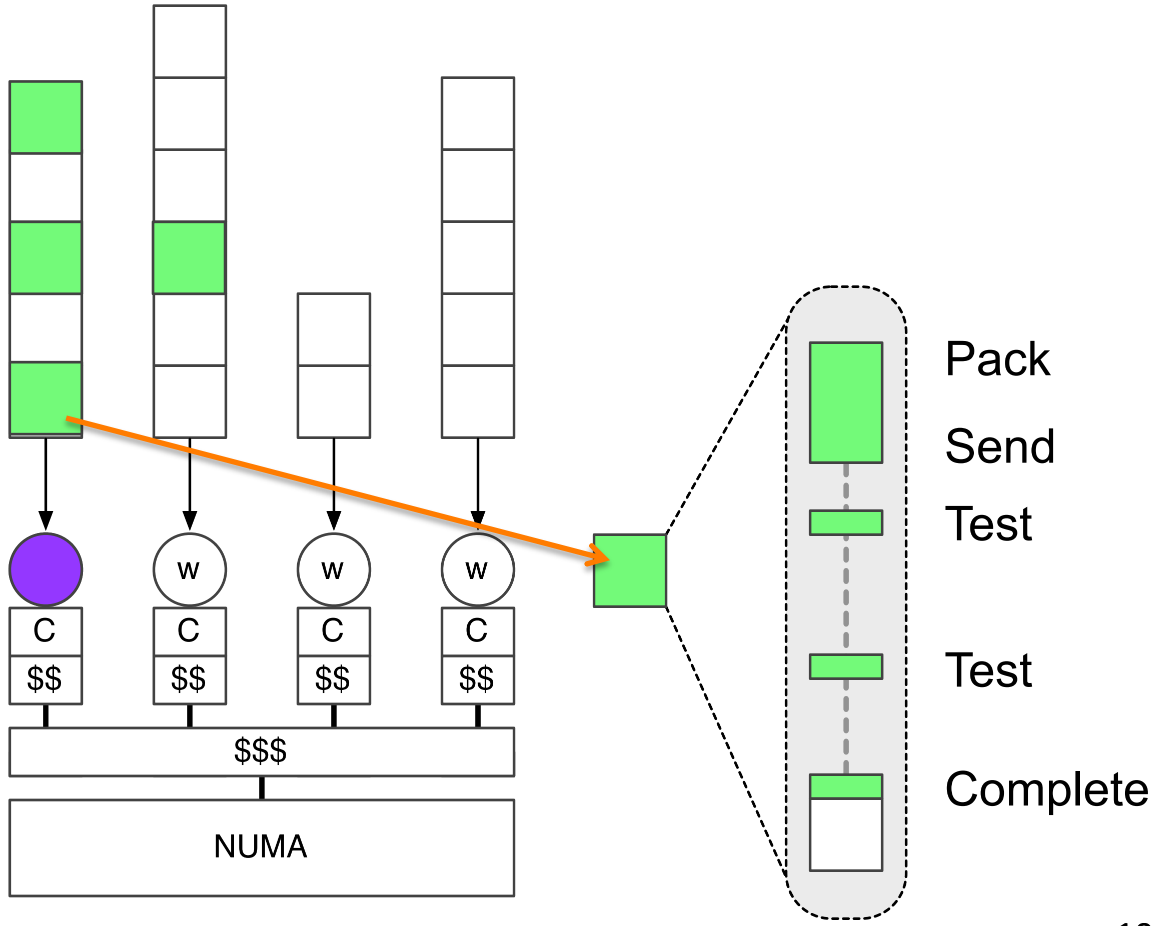
- Ensuring thread safety
- Avoiding deadlock



Design of LLTR

Challenges:

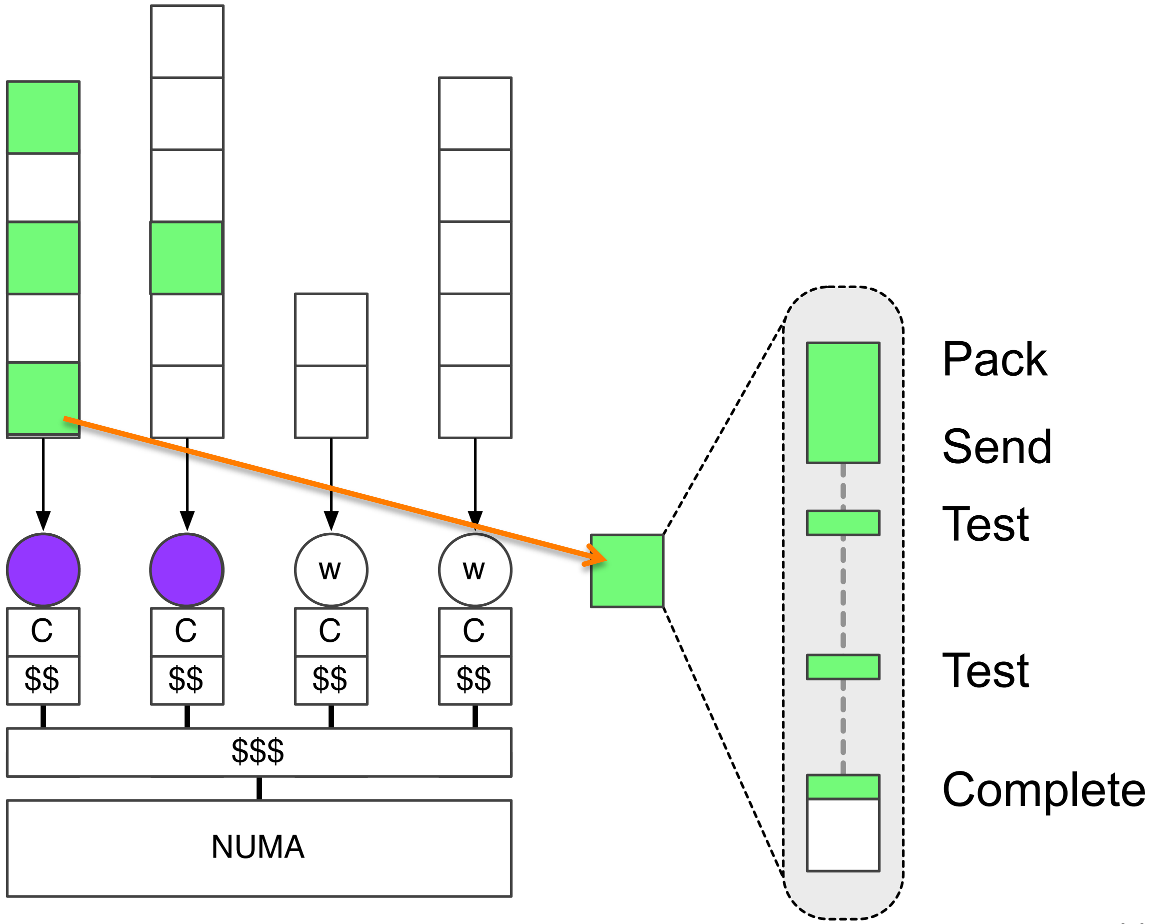
- Ensuring thread safety
- Avoiding deadlock
- Managing long-latency events



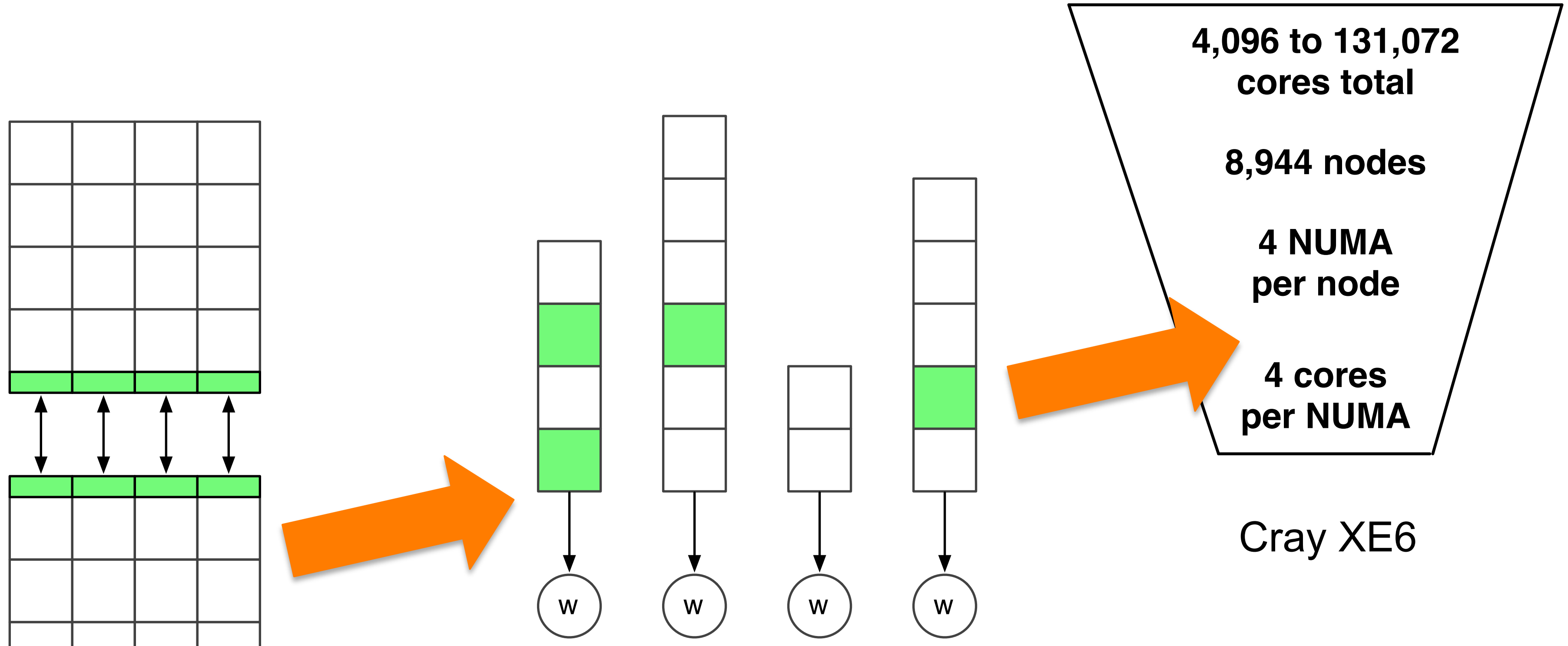
Design of LLTR

Challenges:

- Ensuring thread safety
- Avoiding deadlock
- Managing long-latency events
- Balancing heterogeneous workload and resources



Bringing it all together



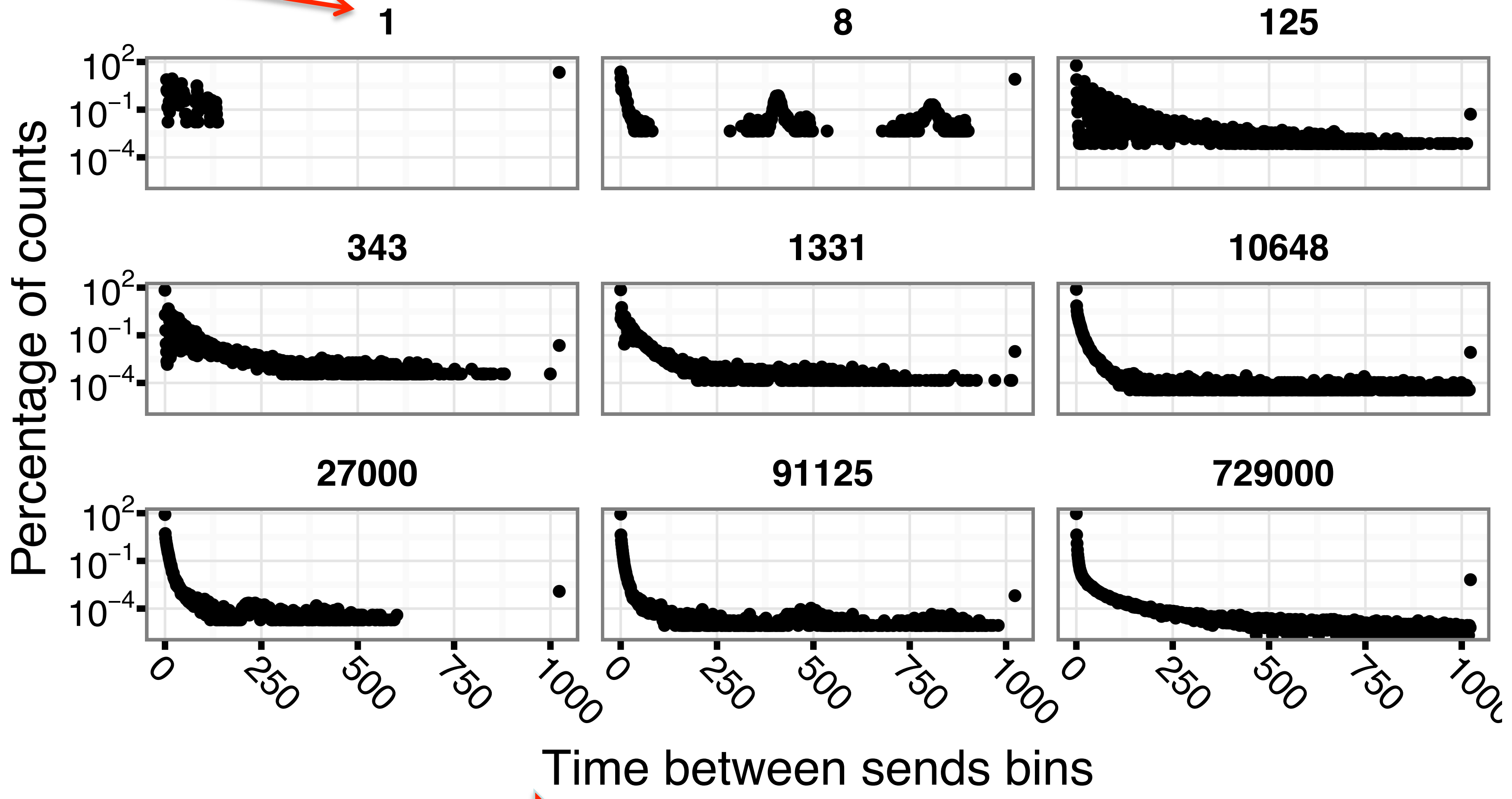
Cray XE6

MPIQ

miniGhost w/ TPOD

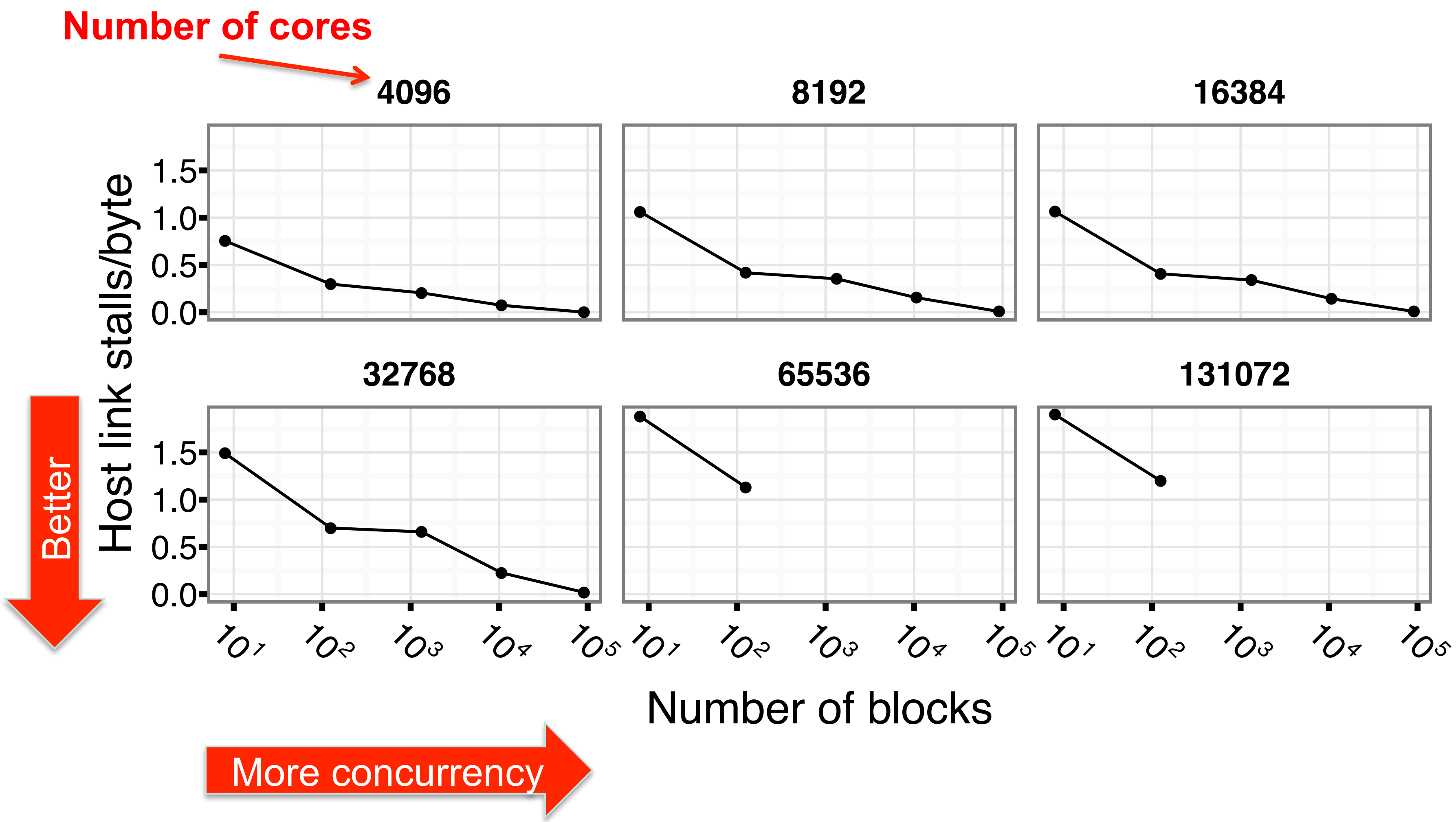
Spreading message injection

Number of blocks

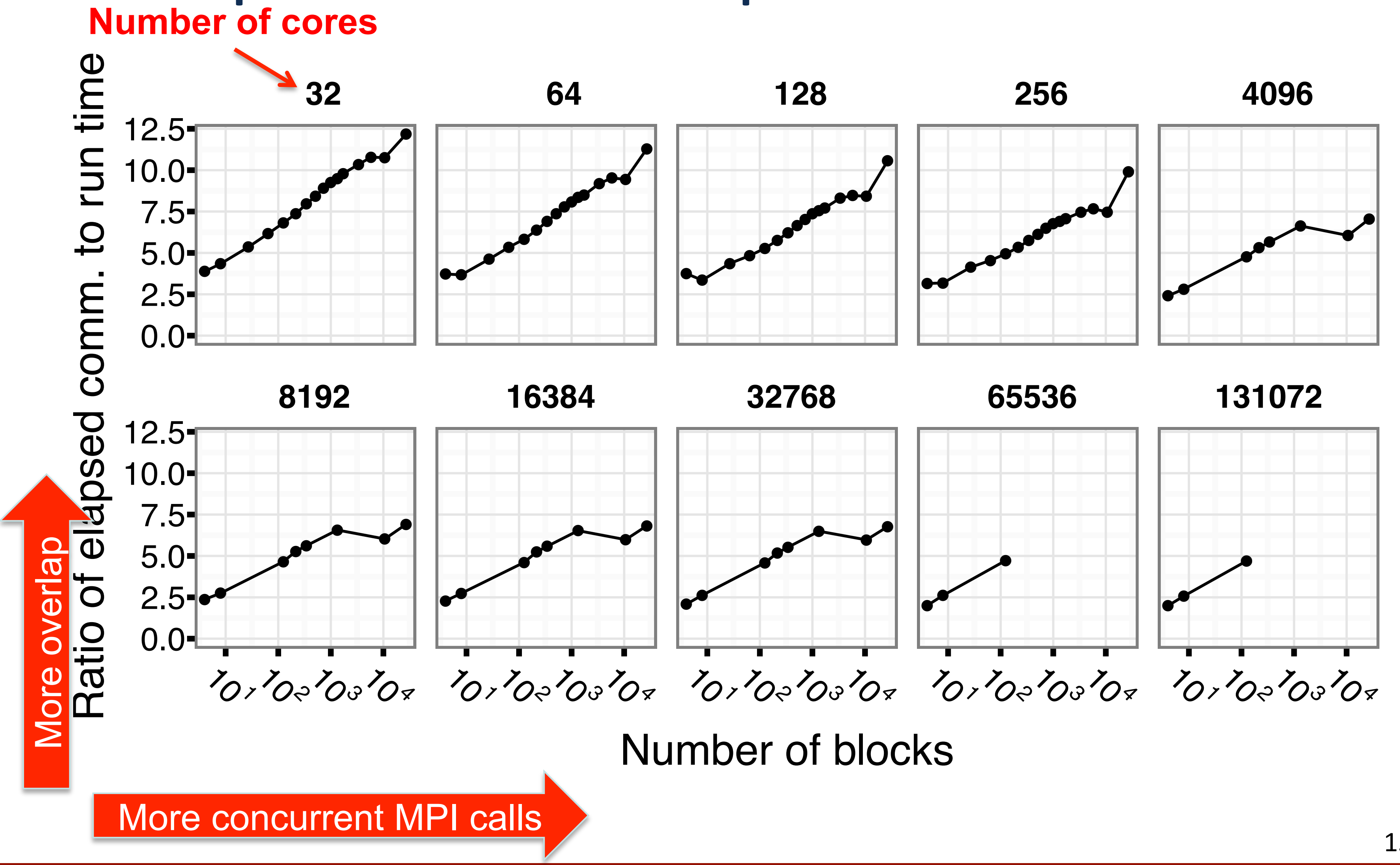


Longer between sends

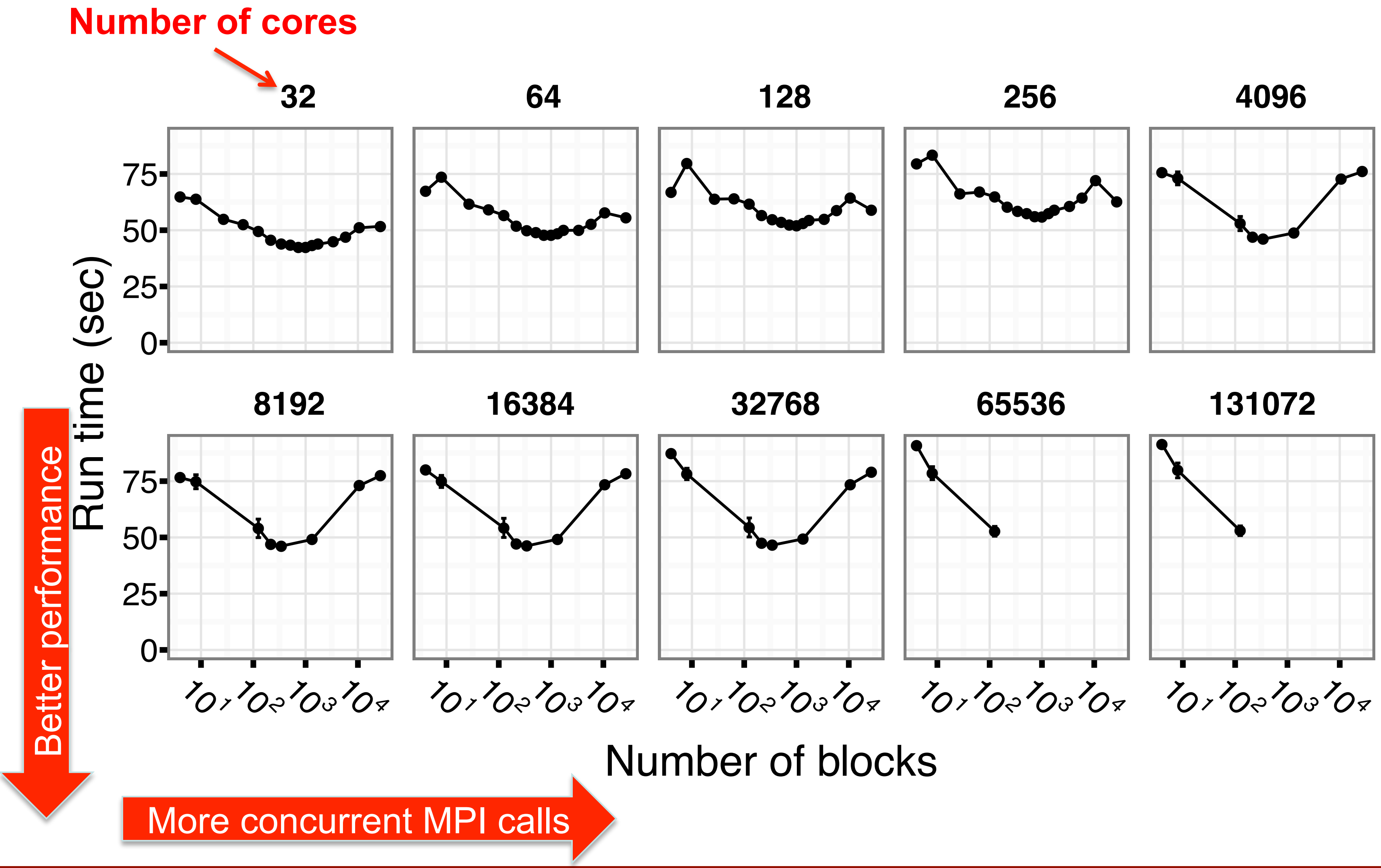
Decreasing network stalls



Increasing communication-computation overlap



Improving performance, to a point



Summary

- Co-schedule work and communication using a task parallel runtime
- Imposed minimal code changes on a proxy application
- Showed promising early results up to 131,072 cores

- Lots more to explore …
  - MPI thread support levels, Barriers, Endpoints, …
  - Interoperability with high-level PMs (OpenMP, Kokkos, …)
  - Runtime support for efficient event-driven wake-up, scheduled polling events, …
  - Use cases for AMR, graph analytics, …