SAND2021-11858C

# Using Monitoring Data to Improve HPC Performance via Network-Data-Driven Allocation

Yijia Zhang[1], **Burak Aksar[1]**, Omar Aaziz[2], Benjamin Schwaller[2], Jim Brandt[2], Vitus Leung[2], Manuel Egele[1] and Ayse K. Coskun[1]

[1]Boston University, USA
[2]Sandia National Laboratories, USA

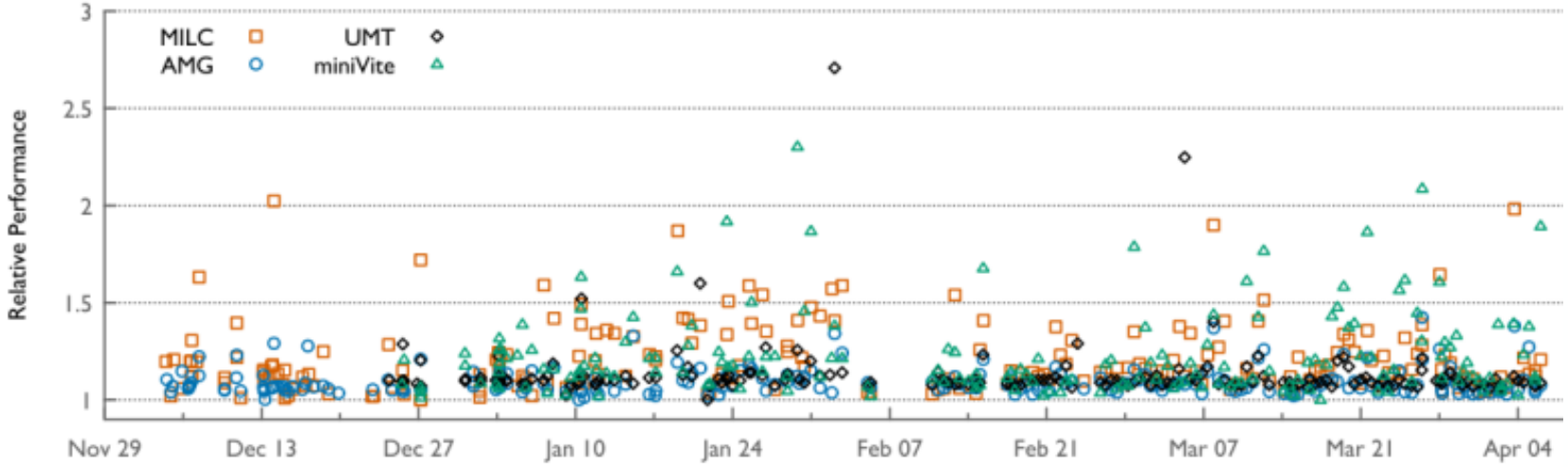
Acknowledgment
This work has been partially funded by Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

# Performance degradation happens on HPC systems

The shared interconnections in a dragonfly system

- Network contention causes HPC performance variability as high as 2x [Bhatele, SC'13], 3x [Bhatele, IPDPS'20], 7x [Chunduri, SC'17], 8x [Zhang, Cluster'20]

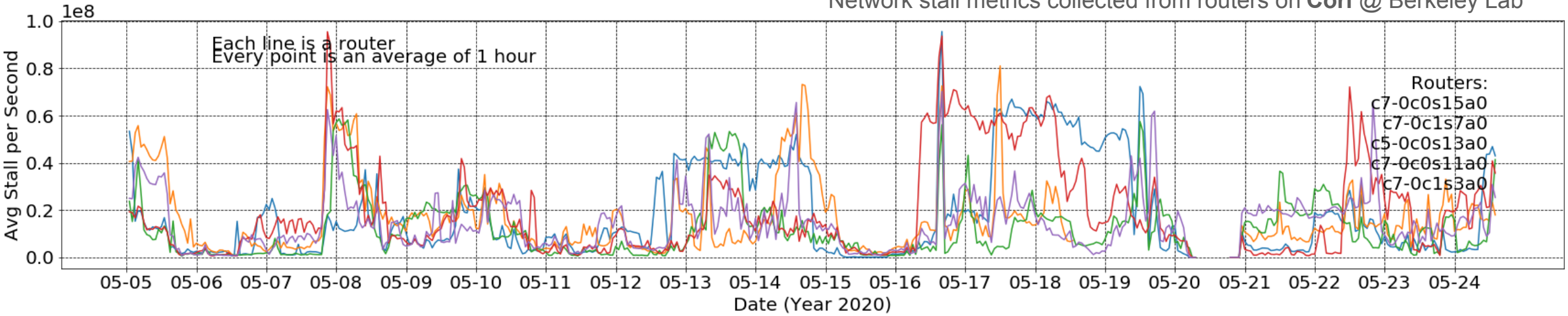
Experiments on **Cori** @ Berkeley Lab [Bhatele, IPDPS'20]



# Monitoring data are available on HPC systems

- The LDMS monitoring system [Agelastos, SC'14] has been deployed on many HPC systems, such as
  - **Cori**, 12k-node, @LBNL, USA
  - Trinity, 19k-node, @LANL, USA
  - Blue Waters, 28k-node, @NCSA, USA

Network stall metrics collected from routers on **Cori** @ Berkeley Lab



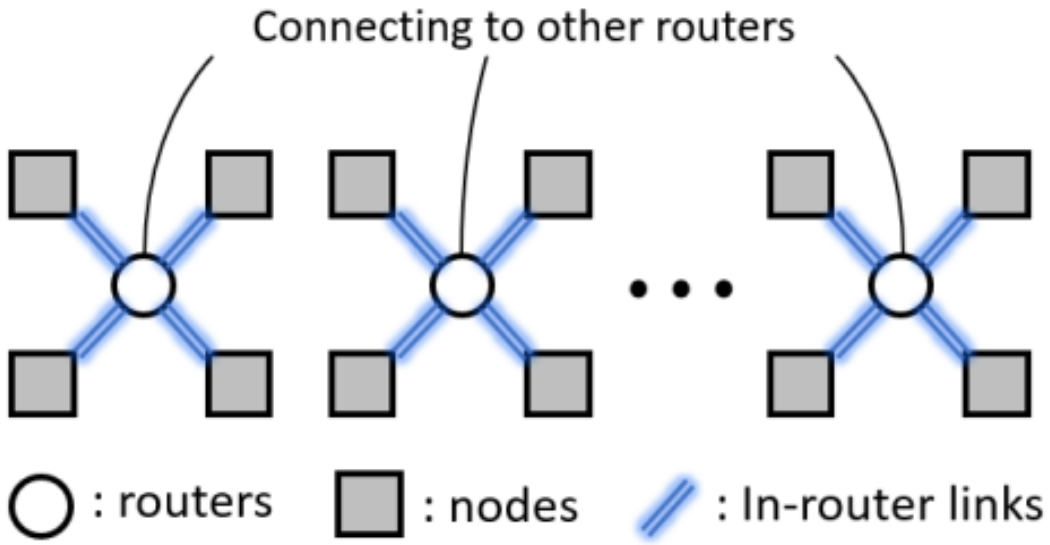
# Related work on HPC job allocation

1. Strategies based on static properties (job size, topology, etc.)
   [Jain, SC'14], [Prisacari, HPDC'14], [Jokanovic, IPDPS'15], [Wang, SC'16]
2. Strategies based on profiled job characteristics (communication graph)
   [Soryani, JoSC'13], [Michelogiannakis, CCGRID'17], [Yan, ICPP'19]
3. Strategies based on thermal and energy constraints
   [Cheng, IPDPS'14], [Cao, IPDPS'17]
4. Strategies based on other running jobs' placement (avoid interference)
   [Pollard, SC'18], [Zhang, IPDPS'18]
5. Strategies based on system performance status / resource utilization
   [Werstein, PDCAT'06], [Yang, JoSC'11], [LaCurts, IMC'13]

- Existing allocation strategies seldom consult system network monitoring data at runtime.
- As a result, these strategies cannot easily adapt to the change of network hot spots.
- Using monitoring data to collect network performance has lower overhead than active network query methods used in earlier works (e.g., LaCurts, IMC'13)

# Network-Data-Driven (NeDD) job allocation framework


- We propose the NeDD job allocation framework
  1. System Network Monitoring: quantify network traffic intensity of routers - Runtime
  2. Application Profiling: determine application's sensitivity to congestion - Offline
  3. Allocation Decision: place job onto nodes by avoiding congestion - Runtime
- Our experiments on Cori shows NeDD reduces application execution time by **11**% on average and up to **34**%

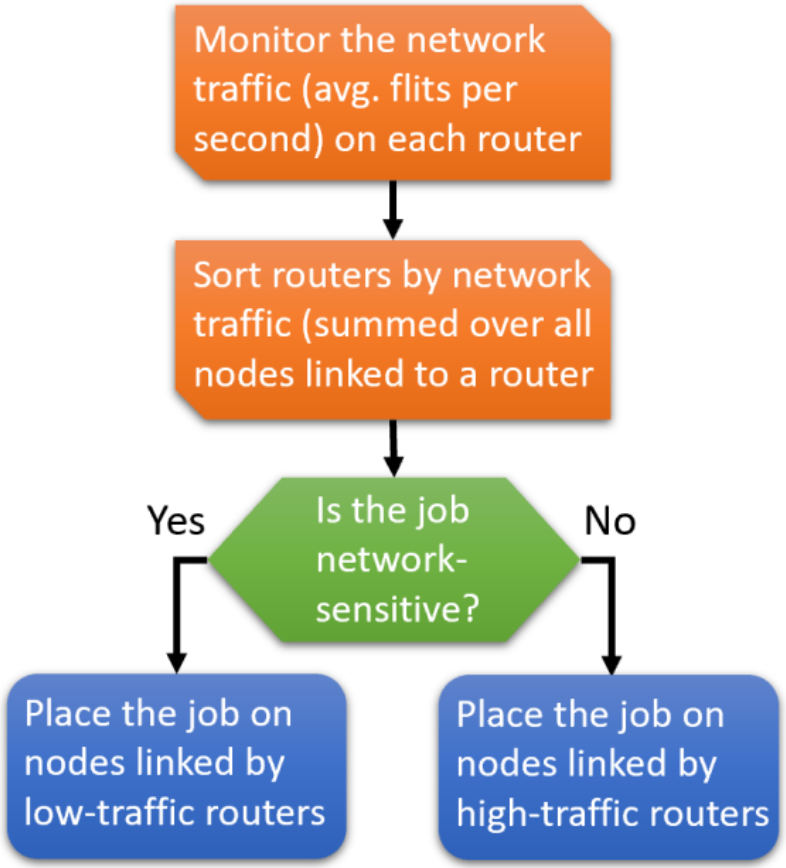




- To quantify the network traffic, we collect the flits per second metric between each node and its parent router
- Define *network traffic intensity* of a router: the total flits per second summed over all nodes directly connected to that router


TABLE I: The ratio of execution time spent on MPI operations.

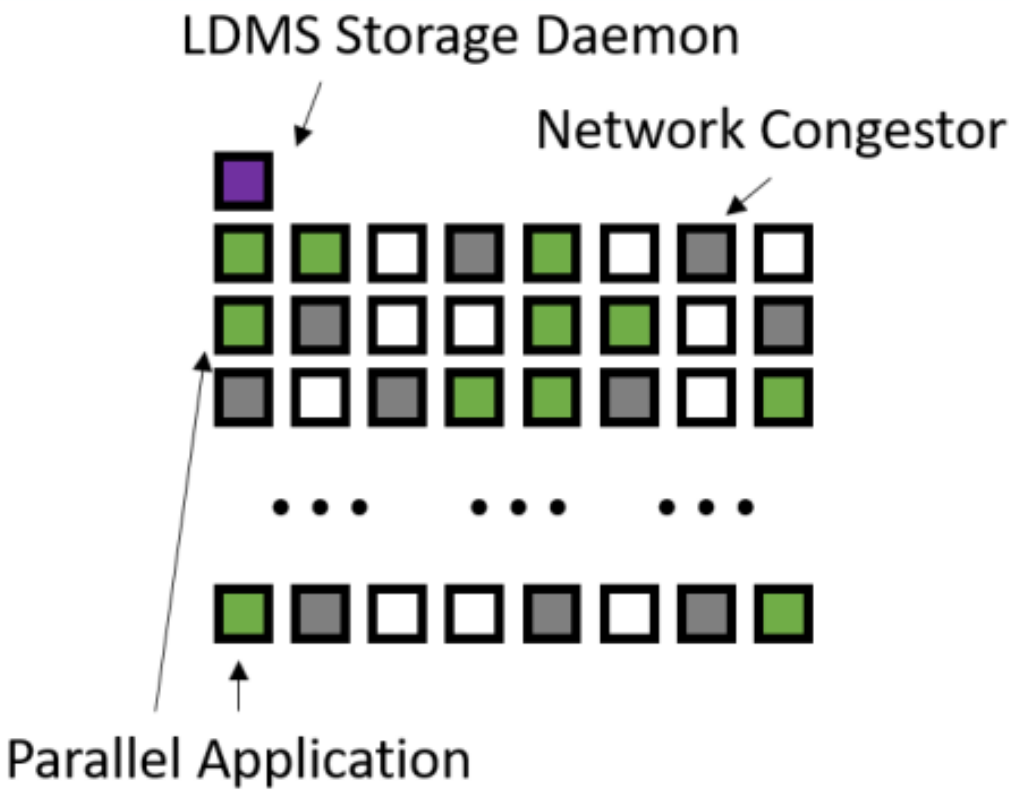
| Application | MPI Operation | MPI_Allreduce | MPI_Sendrecv/Send/Isend | MPI_Wait/Waitall | MPI_(other) |
|---|---|---|---|---|---|
| miniMD | 68.9% | 1.1% | 65.8% | 0 | 2.0% |
| LAMMPS | 51.5% | 22.4% | 10.5% | 12.7% | 5.9% |
| MILC | 48.2% | 1.9% | 7.7% | 34.0% | 4.6% |
| HACC | 49.7% | 1.4% | 0 | 41.2% | 7.1% |
| QMCPACK | 19.3% | 14.2% | 0 | <0.1% | 5.1% |
| HPCG | 11.5% | <0.1% | 4.6% | 6.4% | 0.5% |

- We characterize applications' sensitivity to network congestion based on its MPI statistics collected by the *CrayPat* tool.
  - Our previous work [Zhang, Cluster'20] shows that MPI usage statistics are good indicators of applications' sensitivity to network congestion.
- No need to run an application multiple times under different network conditions to get its sensitivity.


- Allocation Decision:
  - We place a network-sensitive job onto nodes whose parent routers have low network traffic intensity.
  - And vice versa.

# Design of experiments




- Use 201 nodes from the Cori system
  - 64 nodes for Network Congestion
  - 32 nodes for Parallel Application
  - 1 node for LDMS metric storage daemon
- Network congestor: GPCNeT [Chunduri, SC'19]
  - Run an RMA broadcast communication kernel continuously

Step 1: randomly place the network congestor

Step 2: place application following a certain allocation strategy and run

Step 3: repeat with different congestor placement

# Applications in our experiments

| Application | Description |
|---|---|
| HACC | Hardware Accelerated Cosmology Code framework |
| HPCG | High Performance Conjugate Gradient benchmark |
| LAMMPS | Large-scale Atomic/Molecular Massively Parallel Simulator |
| MILC | MIMD Lattice Computation for quantum chromodynamics |
| miniMD | Parallel Molecular Dynamics simulation |
| QMCPACK | Many-body ab initio Quantum Monte Carlo for computing electronic structures |


TABLE I: The ratio of execution time spent on MPI operations.

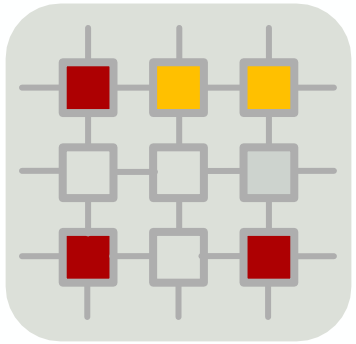
| Application | MPI Operation | MPI_Allreduce | MPI_Sendrecv/Send/Isend | MPI_Wait/Waitall | MPI_(other) |
|---|---|---|---|---|---|
| miniMD | 68.9% | 1.1% | 65.8% | 0 | 2.0% |
| LAMMPS | 51.5% | 22.4% | 10.5% | 12.7% | 5.9% |
| MILC | 48.2% | 1.9% | 7.7% | 34.0% | 4.6% |
| HACC | 49.7% | 1.4% | 0 | 41.2% | 7.1% |
| QMCPACK | 19.3% | 14.2% | 0 | <0.1% | 5.1% |
| HPCG | 11.5% | <0.1% | 4.6% | 6.4% | 0.5% |

Network-sensitive

Network-insensitive

- We determine network intensiveness by looking at time spent on MPI operations

# Job allocation strategies compared

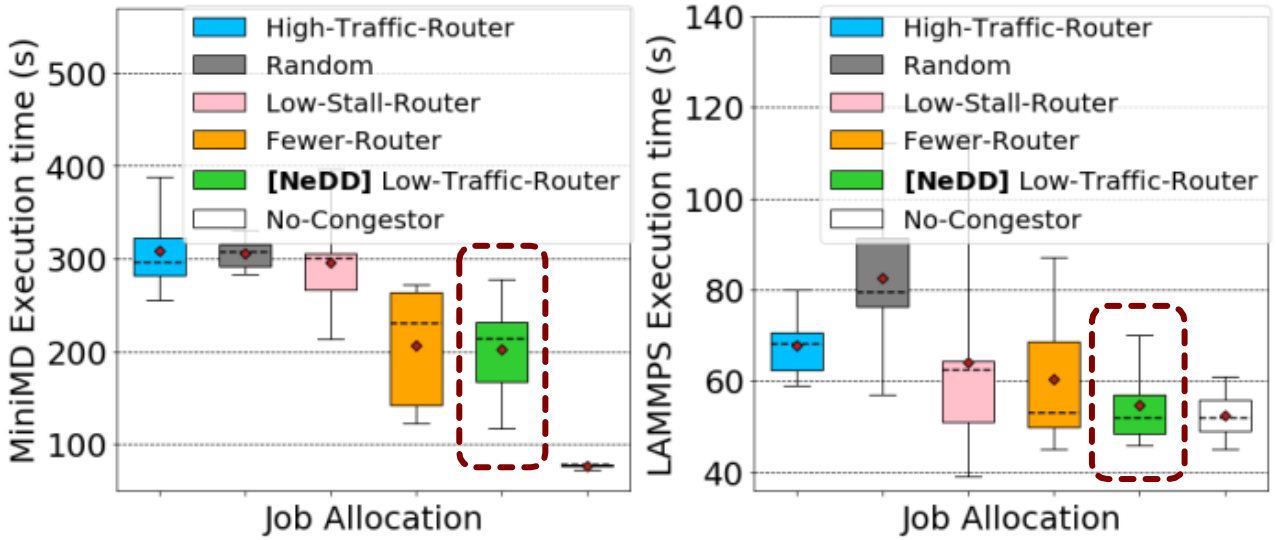


- Low-Traffic-Router places a job on nodes whose parent routers have low network traffic intensity.
  - This is **NeDD's** strategy for network-sensitive applications.
- High-Traffic-Router prioritizes routers with high network traffic intensity.
  - This is **NeDD's** strategy for network-insensitive applications.
- Random strategy places a job randomly.
- Low-Stall-Router strategy prioritizes routers with low Ntile network stalls (i.e., network stall count summed over all Ntiles in a router).
- Fewer-Router places a job into fewer routers by prioritizing routers connected to more idle nodes.

# Experiment results – single job

- For network-sensitive apps, NeDD reduces job execution time by up to 34% when compared with the Random allocation strategy.
- For network-insensitive apps, the performance degradation of NeDD is small.

network-sensitive apps

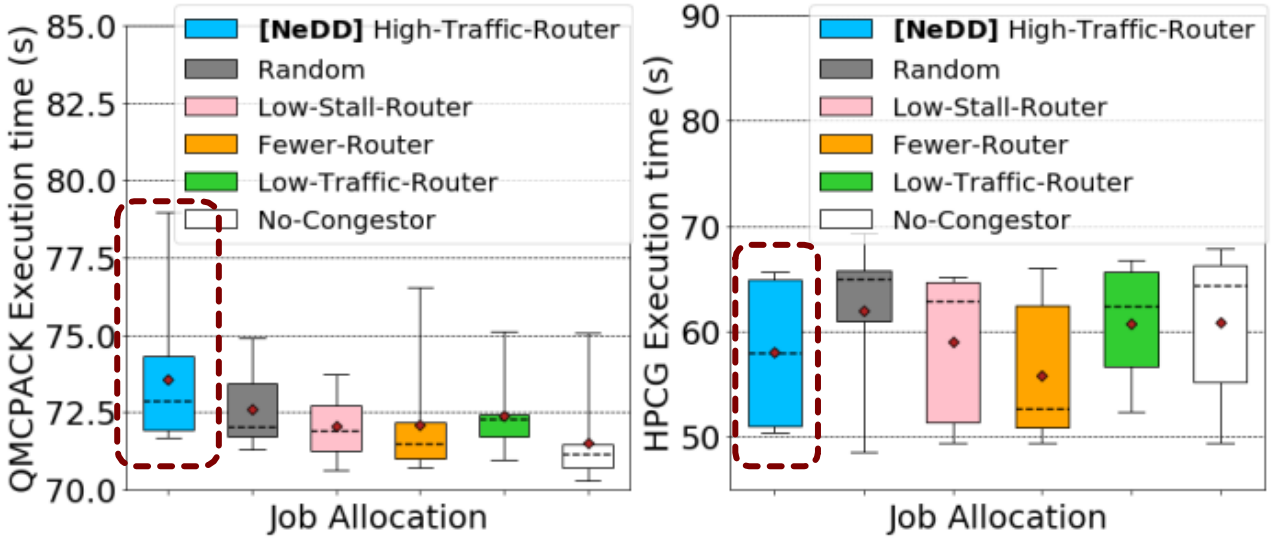



Red point: mean
Dashed line: median

# Experiment results – single job

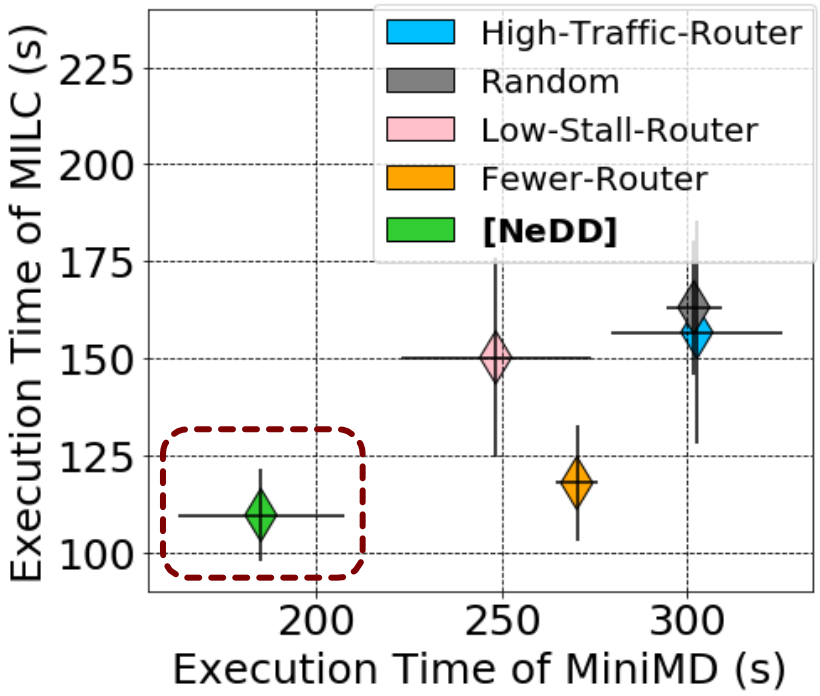
- For network-sensitive apps, NeDD reduces job execution time by up to 34% when compared with the Random allocation strategy.
- For network-insensitive apps, the performance degradation of NeDD is small.

network-insensitive apps




Red point: mean
Dashed line: median

# Experiment results – two jobs

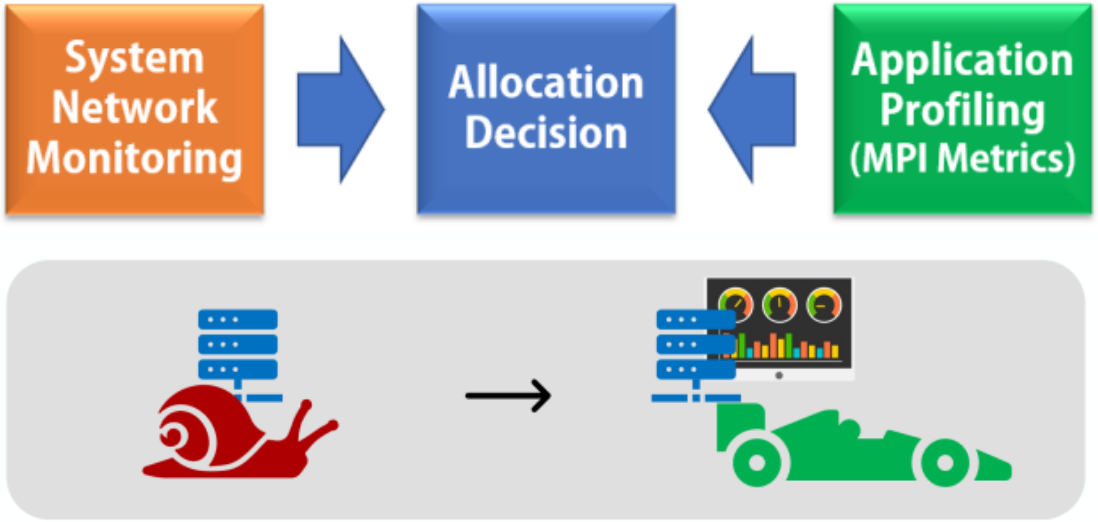



- In this experiments, the congestor is first placed, then two jobs (miniMD and MILC) are placed following a certain allocation strategy, and two jobs start simultaneously.
- X-/Y-axis represents miniMD/MILC's average execution time.
- NeDD allocation strategy improves the performance of both jobs.

# Thank you

Using Monitoring Data to Improve HPC Performance via Network-Data-Driven Allocation

- We demonstrate system monitoring data could be used in job allocation to improve HPC performance.
- NeDD allocation framework could reduce job execution time by up to 34%.




This work is partially funded by Sandia National Laboratories.
This research used resources of NERSC at Lawrence Berkeley National Laboratory.

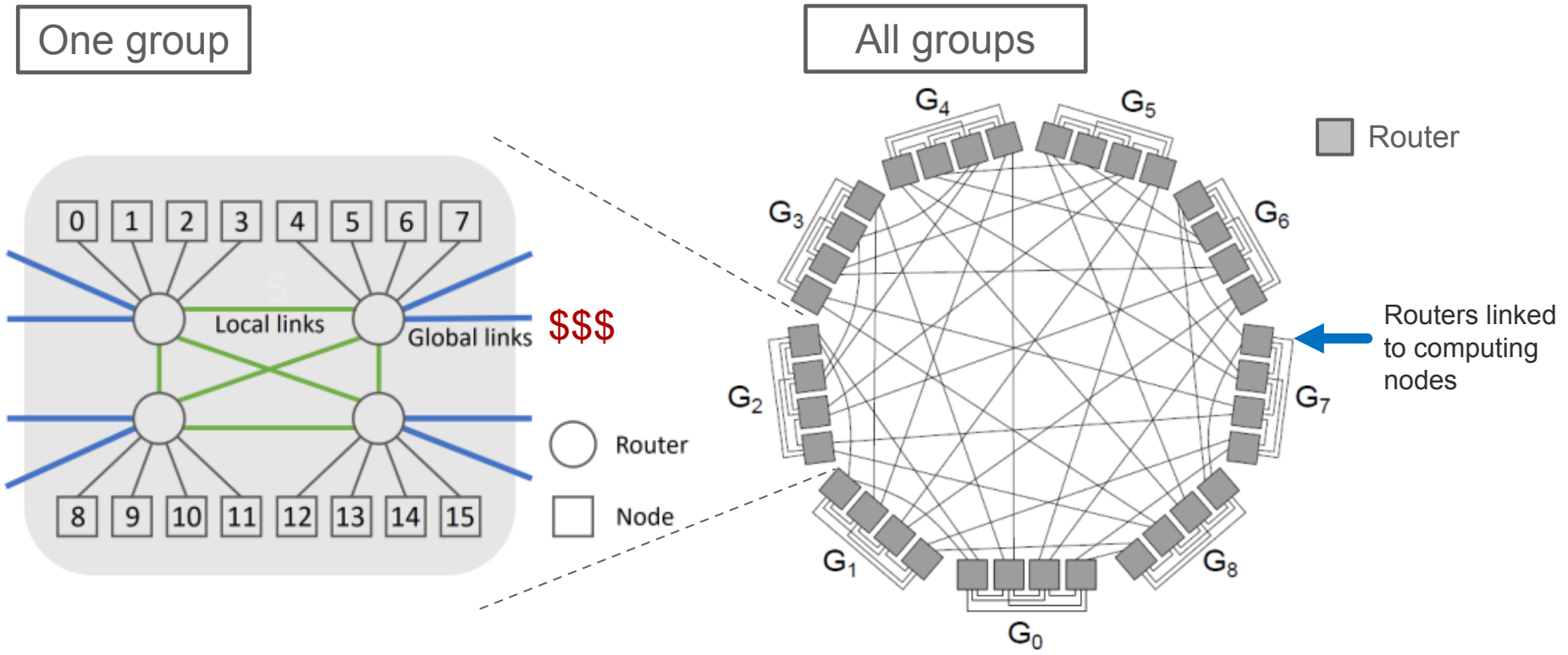
BOSTON UNIVERSITY

Sandia National Laboratories

# BACK UP

# Dragonfly network topology

[Kim, SIGARCH'08]




## Advantages

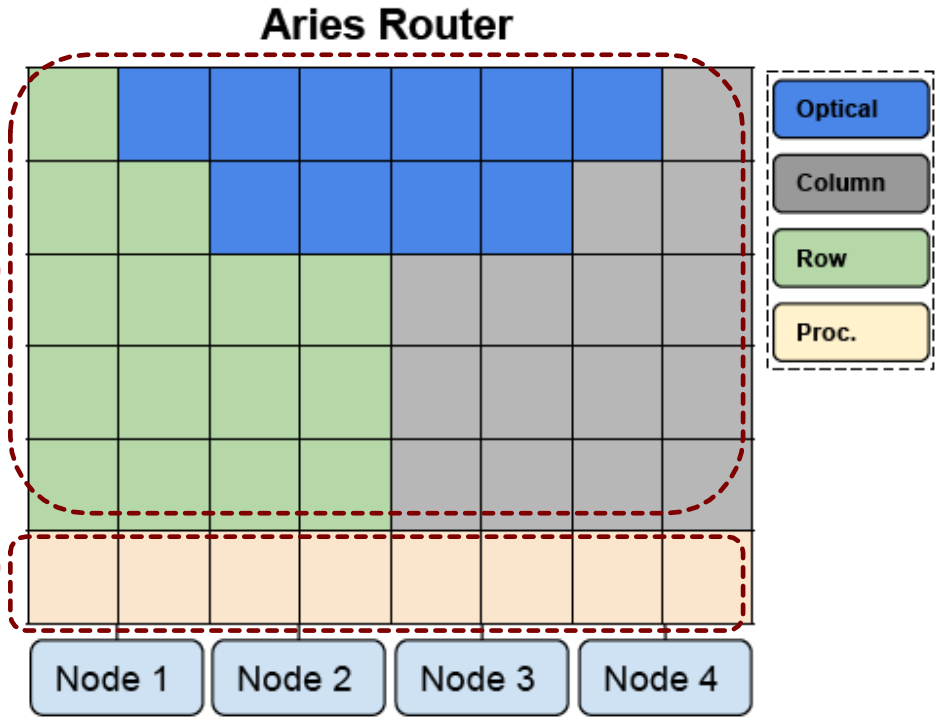
- Constant network diameter
- Employs both electrical & optical network connections

## Implemented in

- Cray Cascade system [Faanes, SC'12]
- IBM PERCS architecture [Chakaravarthy, HiPC'12]

Cori @ NERSC
Edison @ NERSC
Trinity @ LANL
Shaheen II @ KAUST
… …

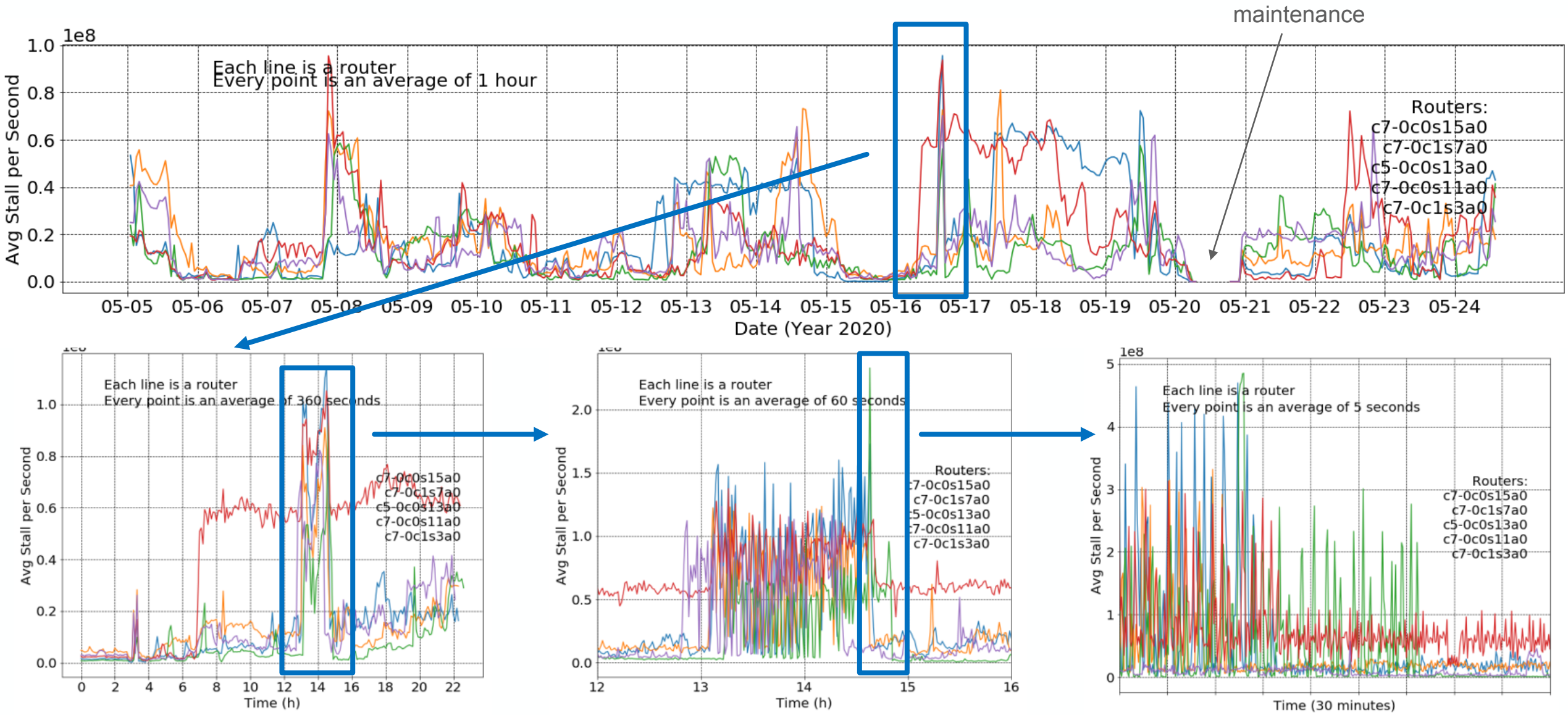
# Aries router counters



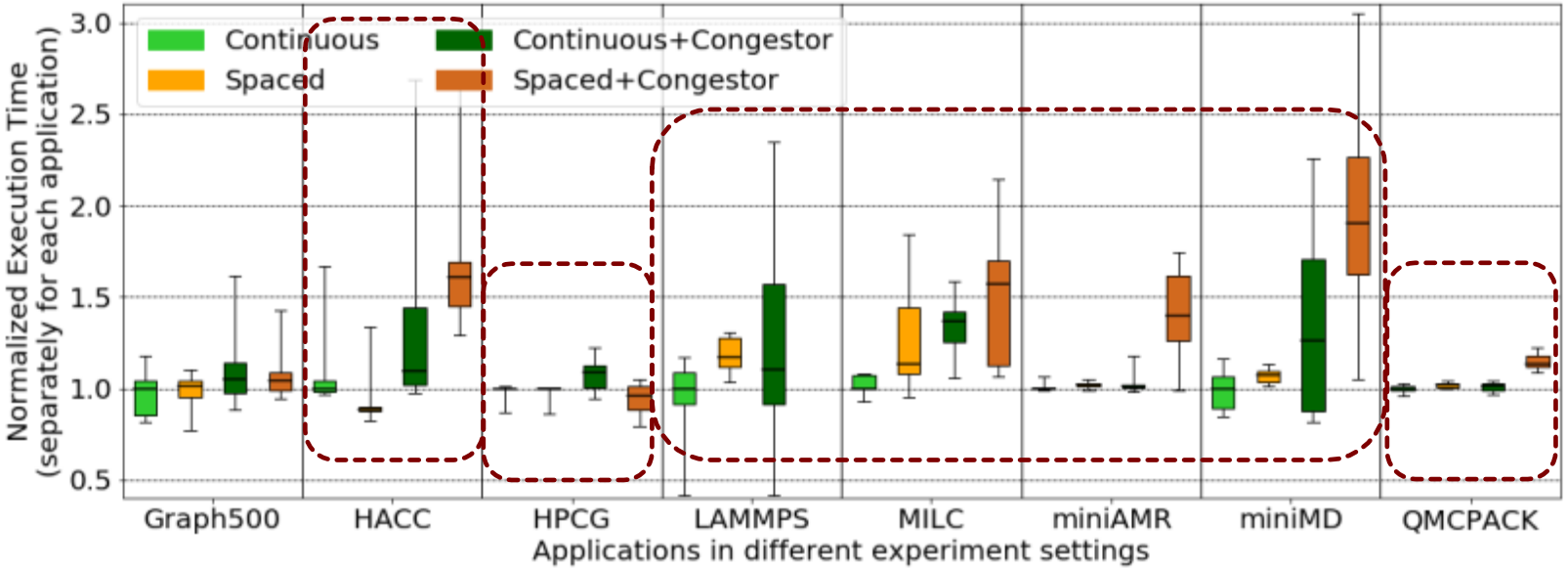

| Abbreviation | Full counter name |
|---|---|
| N_STALL_$r$_$c$ | AR_RTR_$r$_$c$_INQ_PRF_ROWBUS_STALL_CNT |
| N_FLIT_$r$_$c$_$v$ | AR_RTR_$r$_$c$_INQ_PRF_INCOMING_FLIT_VC$v$ |
| P_REQ_STALL_$n$ | AR_NL_PRF_REQ_PTILES_TO_NIC_$n$_STALLED |
| P_REQ_FLIT_$n$ | AR_NL_PRF_REQ_PTILES_TO_NIC_$n$_FLITS |
| P_RSP_STALL_$n$ | AR_NL_PRF_RSP_PTILES_TO_NIC_$n$_STALLED |
| P_RSP_FLIT_$n$ | AR_NL_PRF_RSP_PTILES_TO_NIC_$n$_FLITS |

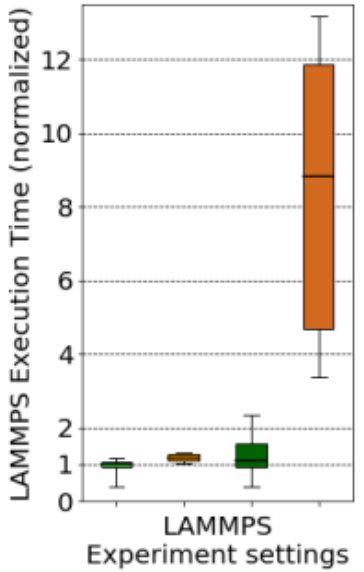
# Analyze network stall-per-second on Cori

Network congestion sometimes stays in the system for hours/days.



# Relation with MPI profiles

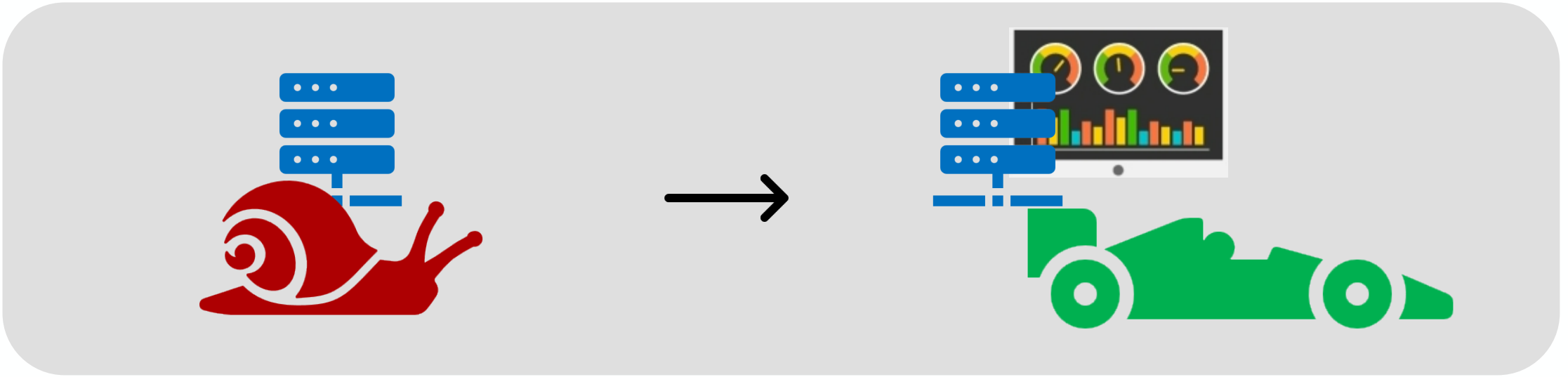



| Application | MPI Operation | MPI_Allreduce | MPI_Sendrecv (or Send, Isend) | MPI_Bcast | MPI_Test (or Testany) | MPI_Wait (or Waitall) | MPI_Barrier | MPI_(other) |
|---|---|---|---|---|---|---|---|---|
| Graph500 | 31.4% | 2.3% | 2.6% | 0.2% | 21.3% | <0.1% | 4.4% | 0.6% |
| HACC | 67.1% | <0.1% | 0.2% | 0 | 0 | 66.2% | 0 | 0.7% |
| HPCG | 3.7% | 0.2% | 2.1% | 0 | 0 | 1.2% | 0 | 0.2% |
| LAMMPS | 47.3% | 25.4% | 8.6% | <0.1% | 0 | 12.2% | <0.1% | 1.1% |
| MILC | 61.9% | 1.9% | 0.6% | <0.1% | 0 | 58.5% | <0.1% | 0.9% |
| miniAMR | 26.8% | 9.2% | 0.5% | <0.1% | 0 | 14.2% | 0 | 2.9% |
| miniMD | 83.4% | 0.5% | 82.5% | 0 | 0 | 0 | 0.2% | 0.2% |
| QMCPACK | 6.0% | 5.4% | <0.1% | <0.1% | <0.1% | <0.1% | 0.5% | 0.1% |

Profiled by CrayPat

- HACC, LAMMPS, MILC, miniAMR, miniMD are high in MPI operation time
- LAMMPS and miniAMR are high in MPI collective operations
- HPCG and QMCPACK are low in MPI operation



Improve HPC performance using monitoring data