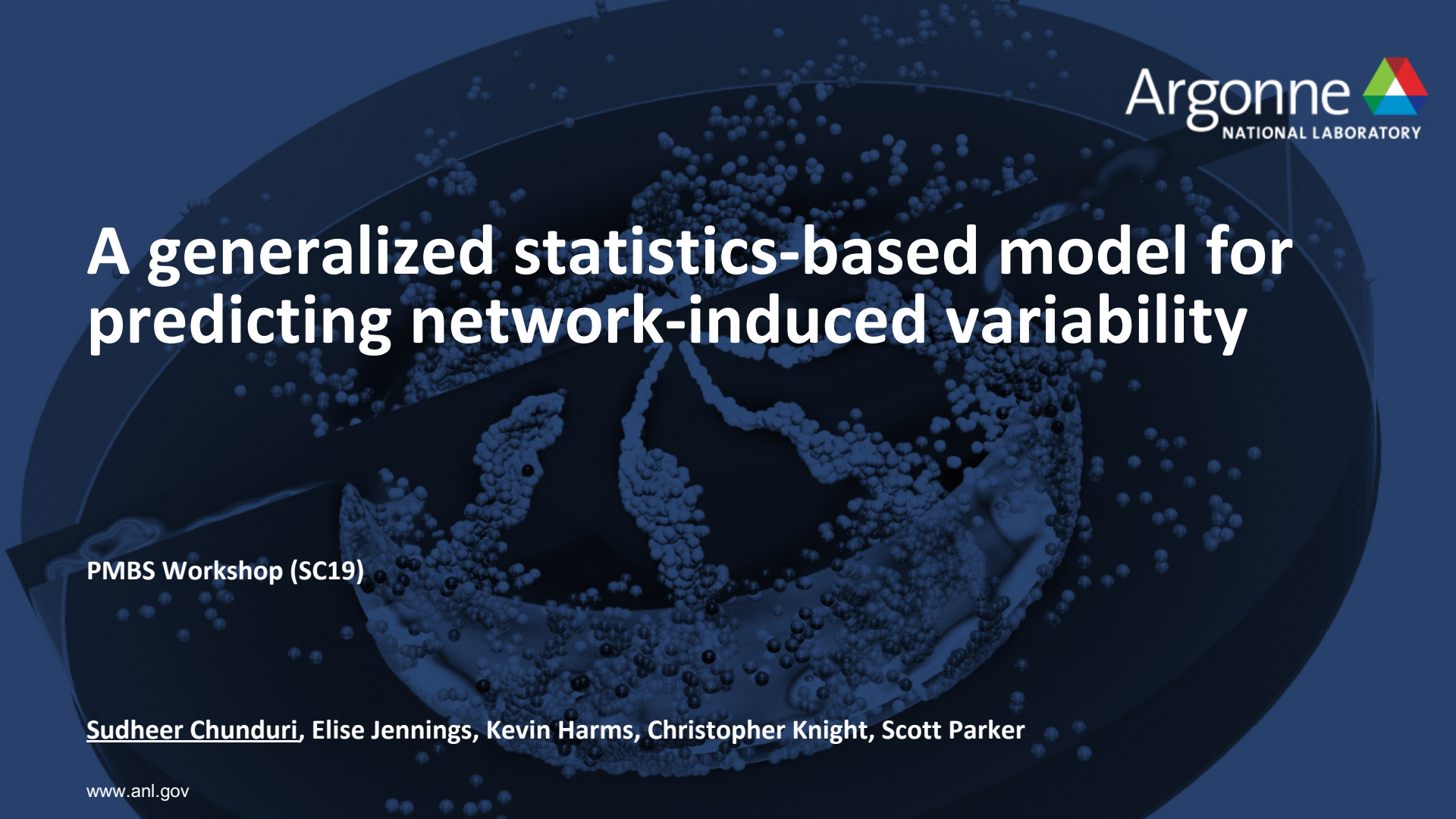
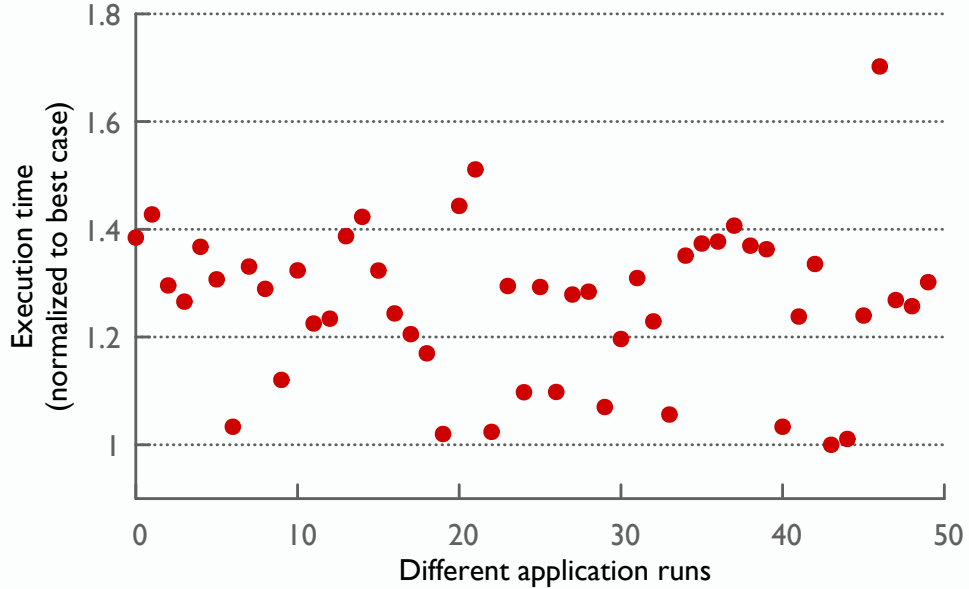


A generalized statistics-based model for predicting network-induced variability

PMBS Workshop (SC19)

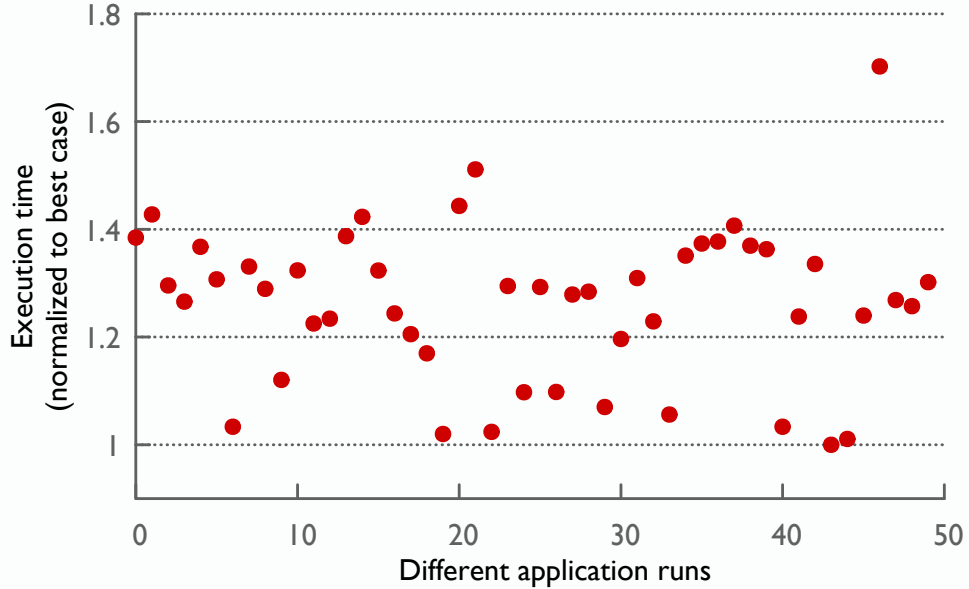
Sudheer Chunduri, Elise Jennings, Kevin Harms, Christopher Knight, Scott Parker

Application Run-to-run Variability



Production application (MILC) run-to-run variability on Cray XC40 (Theta)

Application Run-to-run Variability

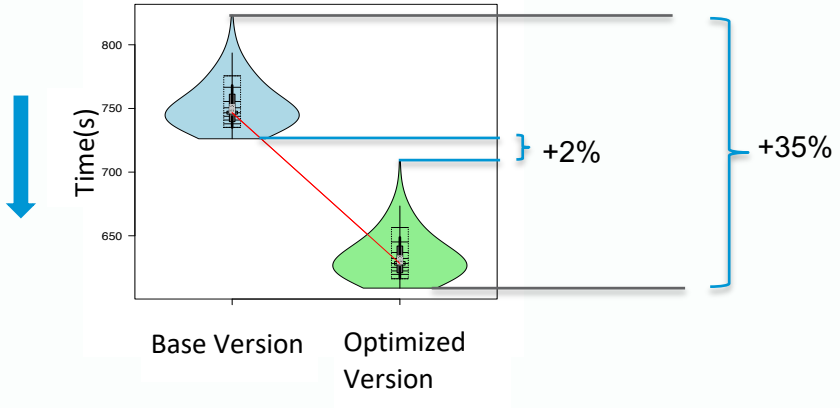


Production application (MILC) run-to-run variability on Cray XC40 (Theta)

- **Variation caused by interference from other running jobs**
- **The interference seen is a function of**
  - ***System "state" specific parameters (node placement, job size, routing etc.)***
  - ***Application specific aspects (communication intensity, pattern etc.)***

Issues with Performance Variability

- Accurate performance benchmarking is not straightforward
- Less reliable performance measures requiring **multiple repetitions** (unknown number of repetitions)
- Statistical significance analysis is required for presenting the performance data
- Performance tuning analysis under these environments could be misleading
  - The range of variability could be relatively higher compared to the quantum of performance tuning benefit



Base version of MILC and an optimized version of MILC are compared:

- What is the performance benefit from the tuning optimization?
  - Is it 2% or 35% ?
- Even more challenging if the distributions overlap?

Just having the mere runtime will not be enough for accurately assessing

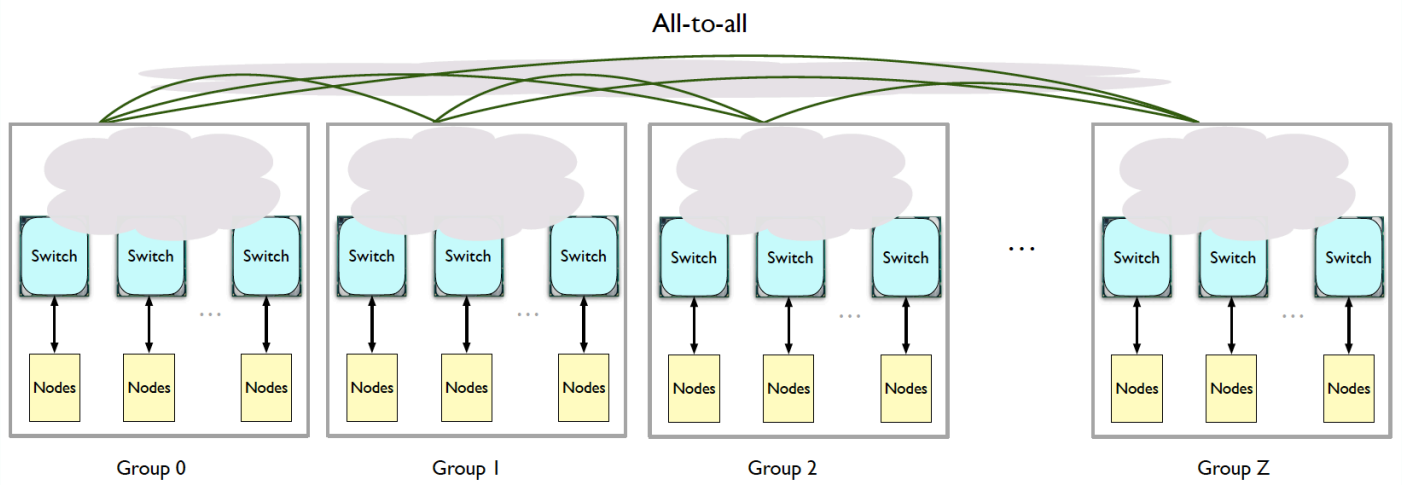
- performance tuning benefit
- scaling efficiency
- etc.

Our Proposed Solution

- We argue that having Runtime + “Context” is essential to accurately interpret performance
- “Context”: the factors that can potentially affect an application from having the best possible runtime
  - System specific factors : congestion on the network due to inter-job contention, node placement etc.
  - Application specific : communication intensity, communication noise sensitivity etc.
- We propose to capture the Context for each application run using the local network performance counters
  - Introduce the network topology used and the network performance counters
  - Derive metrics based on the counters
  - Show the correlation of metrics and the runtime variability
  - Development of a predictive model

Dragonfly Topologies

- High-radix, low diameter and highly scalable network topology
- Requires global adaptive routing and advanced congestion look ahead for efficient operation



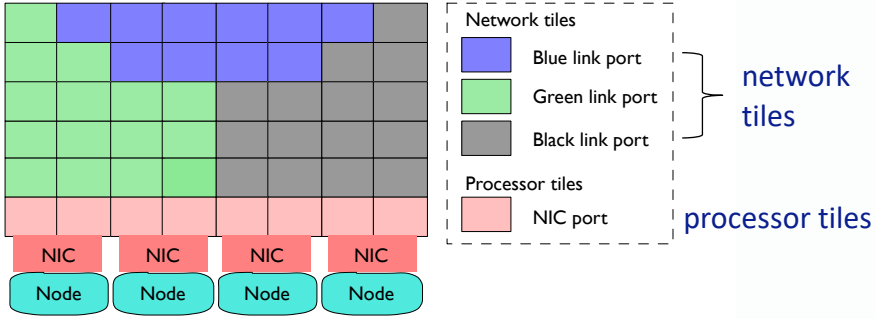
Variability on Dragonfly based networks

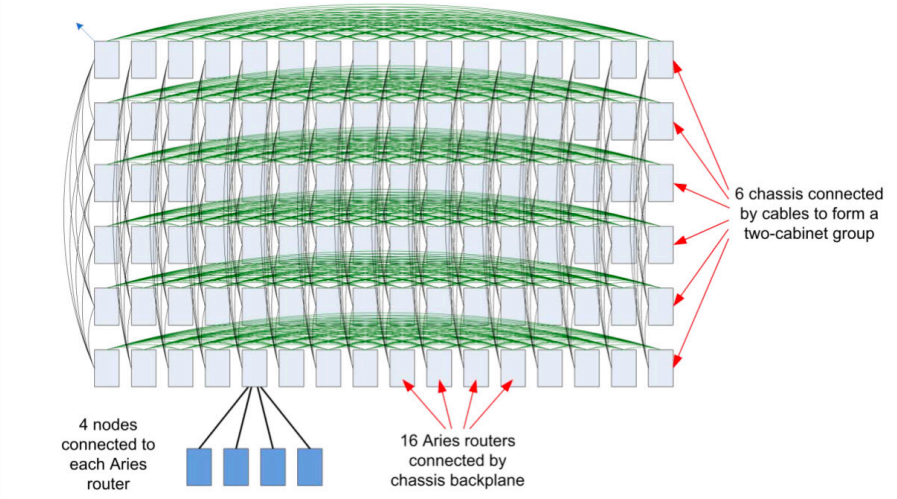
- System design optimized for overall system throughput
- Up to *70%* variability for some applications on *Theta*
- ***Inter-job interference and resulted network congestion*** is the root cause
- No obvious mitigation strategies known to remove the congestion

Cray XC40 Network

- **Aries interconnect: combination of Hamming graph & dragonfly**
- **Router radix = 48 ports**
  - node: 8 (2 ports per node) – processor tiles
  - rank-1: 15 (Black link ports)
  - rank 2: 15 (3 green ports per connection)
  - rank 3: 10 (global Blue ports)

  } network tiles
- **Rank 1: complete graph of 16 routers**
  - 16 Aries, 64 nodes
- **Rank 2: group of 6 rank-1 graphs; Hamming graph K16 x K6**
  - 96 Aries, 384 nodes
- **Up to 96x10+1 = 961 groups; in practice limited to 241 groups**
  - 23,136 Aries, 92,544 nodes
- **Theta**
  - 12 groups, 3 global links between every two groups
  - 4392 nodes





Network Performance Counters on Cray Aries

- Counters used - FLITs and STALLs
  - FLITs: record the atomic unit of data transfer
  - STALLs: incremented each FLIT time that a ready-to-forward FLIT is unable to forward
  - STALLs are an indication of the **back pressure** limiting the rate at which flits are forwarded.
  - A high ratio of STALLs to FLITs indicates possible network congestion
- STALLs and FLITs from 40 network tiles and 8 processor tiles are recorded
- The routers can be shared by other running jobs depending on the node placement
  - request/response traffic from both application and other jobs
  - Network tiles: represent the global traffic occurring on the system from a local application viewpoint
  - Processor tiles: the application-specific characteristics that are unique to that application

Performance Counter-based Metrics to construct the *Context*

- Six metrics are constructed based on the Performance Counters

Parameter	Metric	Description
Application specific	Processor tile flits per second	Application Communication Intensity
Application + Contention	Network tile stall-to-flit ratio	Relative congestion on the network
Application + Contention	Network tile to processor tile flit ratio	Relative communication on the global network vs. the injection rate of the application
Contention	Network stalls	The total stalls, a high absolute value indicates extreme congestion events
Application + Contention	Network flits	The total traffic, an indication of how intense the traffic is
Job size	Nodes	Number of nodes used for a run

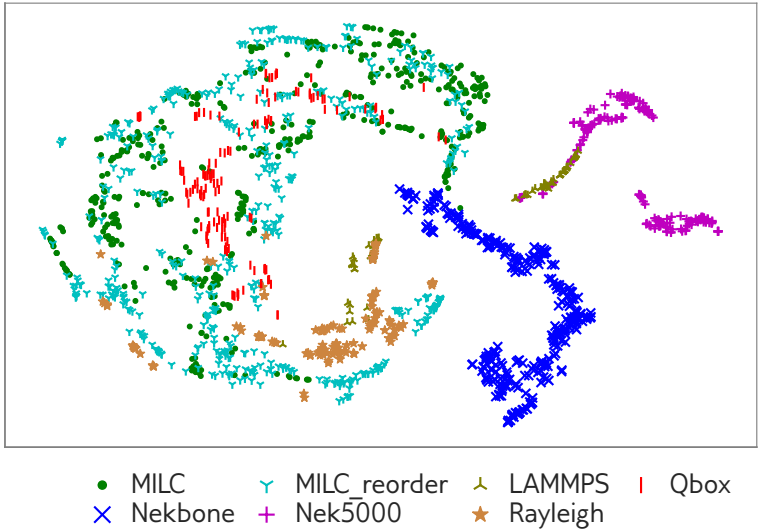
- Mean, Variance, percentiles (75, 95 and 99) for the metrics are used to capture the variability
- A subset of these metrics are selected as ***features*** of importance

Production Applications

- Seven Applications with a wide range of communication patterns are used
- Run at different job sizes (256, 512 and 1024 nodes)
- Ran under different times, thus potentially having
  - Different congestion backgrounds
  - Different node placements
- Variability from sources other than network are avoided

App	Point-to-point	Collectives	Approx. Comp.-to-Comm.Ratio
MILC	heavy	light allreduce	3:2
MILC REORDER	medium	light allreduce	2:1
Nekbone	medium	light	8:1
Nek5000	medium	light	4:1
Qbox	medium	medium	1:2
Rayleigh	none	heavy	2:1
LAMMPS	light	medium	2:1

- t-SNE (t-Distributed Stochastic Neighbor Embedding) is a nonlinear dimensionality reduction algorithm used for mapping high-dimensional space to 2D space
- Feature spaces for Nekbone and Nek5000 are quite distinct from other applications
- Feature spaces for some applications such as MILC, MILC Reorder, Qbox and Rayleigh overlap



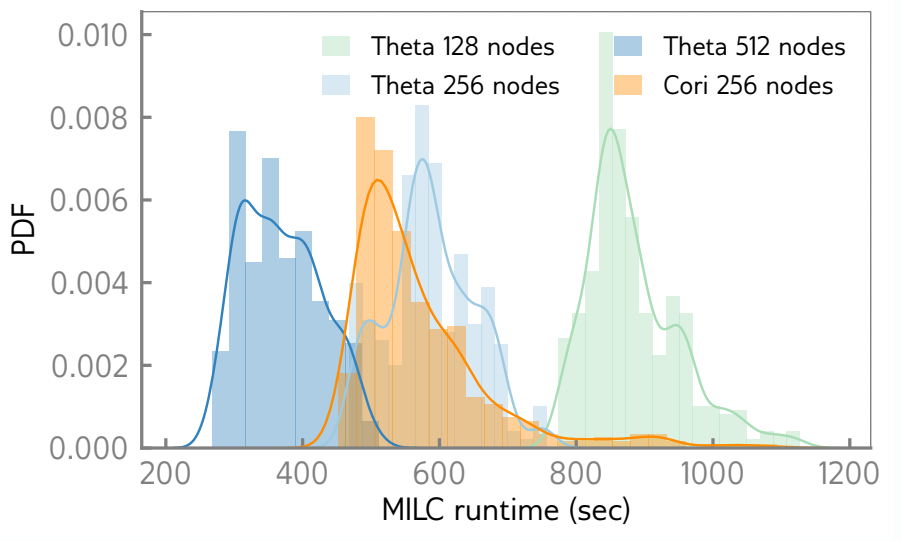
Application Runtime Variability

Runtimes for MILC application runs on 3 different node sizes on Theta
Extended tails can be seen in the runtime distributions
This behavior is consistent across all the applications studied

The maximum runtimes for MILC Reorder, Nekbone and Qbox are also significantly deviated from their mean

This variation is not captured by the CV statistic

Statistics that represent the tails must be used for better characterizing the runtime variability



App	$\mu \pm \sigma$	Max	CV	Max/Min
MILC	566.2 ± 61.90	771.8	0.11	1.71
MILC Reorder	528.7 ± 57.03	699.5	0.11	1.74
Nekbone	1585.1 ± 101.8	1960.5	0.06	1.41
Nek5000	429.9 ± 12.16	472.3	0.03	1.13
Qbox	675.8 ± 43.90	824.4	0.07	1.39
Rayleigh	680.9 ± 18.68	763.5	0.03	1.18
LAMMPS	721.6 ±25.50	779.5	0.04	1.12

Features of Importance and their Characteristics

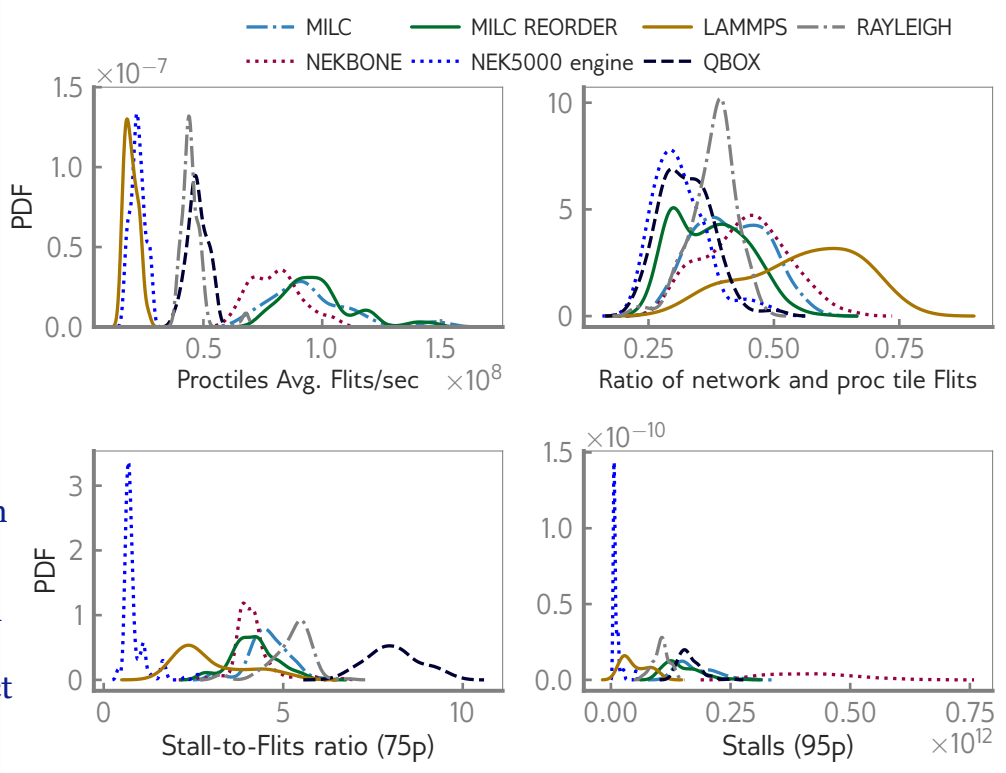
Shows the distribution of four key metrics for all the applications studied

Processor (NIC) tiles defines the communication rate for the application as measured from NIC.

MILC, Reorder are similar.
Apps show distinct characteristics.

The relative congestion seen on all network tiles in tail of the distribution.

The tail of the distribution emphasizes the extreme congestion that can impact the application synchronization stages.



Gives insight into the relative network utilization between the target application and all other applications running on the system during the same time period.

The 95th percentile of stalls (bottom-right) is an absolute measure of conditions where network data is ready, but no network resources are available to send.

Nekbone has very long tails

Correlation of Runtime and Performance Counter-based Metrics

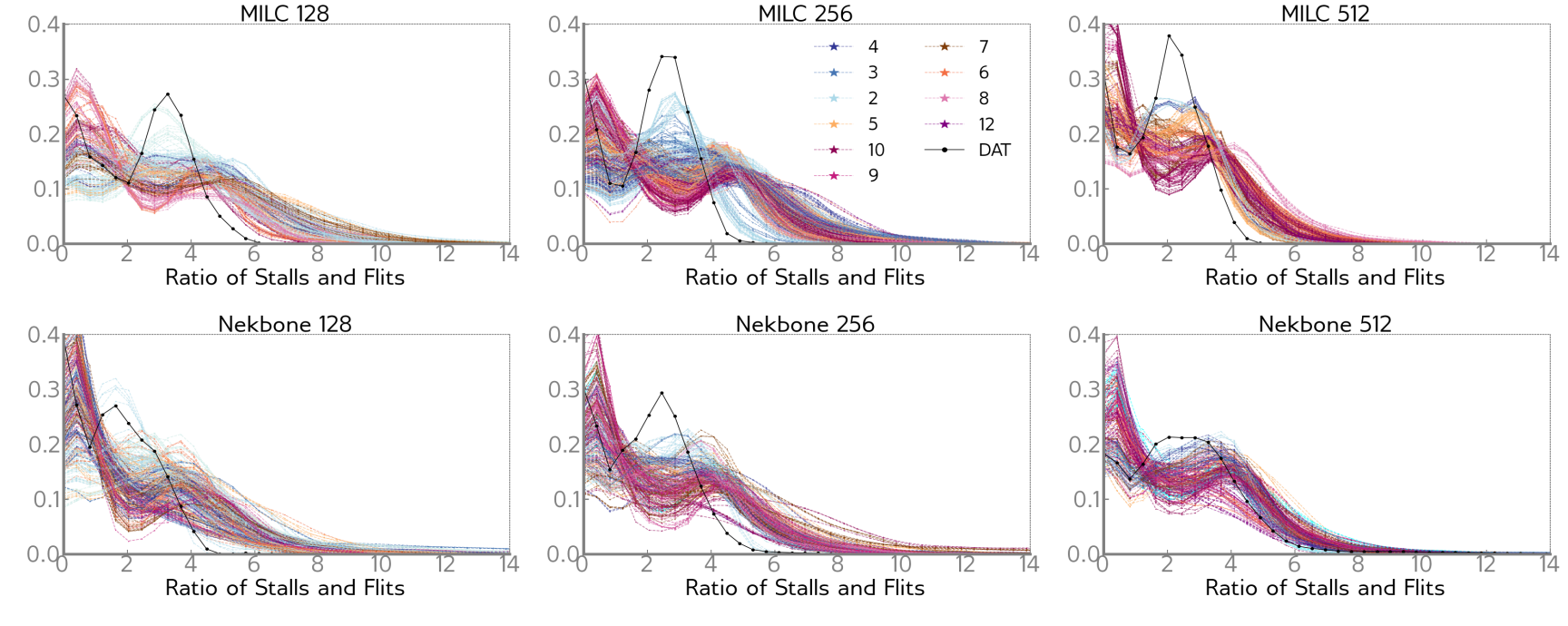
- **Pearson correlation coefficient**, *r*, was used as a measure of the degree of correlation between an application's runtime and a network counter metric
- MILC, Reorder, Nekbone, Qbox and LAMMPS applications there is at least one metric that has an $r > 0.6$
- Nek5000 and Rayleigh do not show any significant correlation

App	Ratio of Flits on networktiles and proctiles	Ratio of Stalls and Flits	Stalls (95 percentile)
MILC	0.65	0.69	0.79
Reorder	0.69	0.67	0.79
Nekbone	0.53	0.42	0.64
Nek5000	0.10	0.29	0.17
Qbox	0.61	0.17	0.72
Rayleigh	0.19	0.32	0.32
LAMMPS	0.56	0.84	0.76

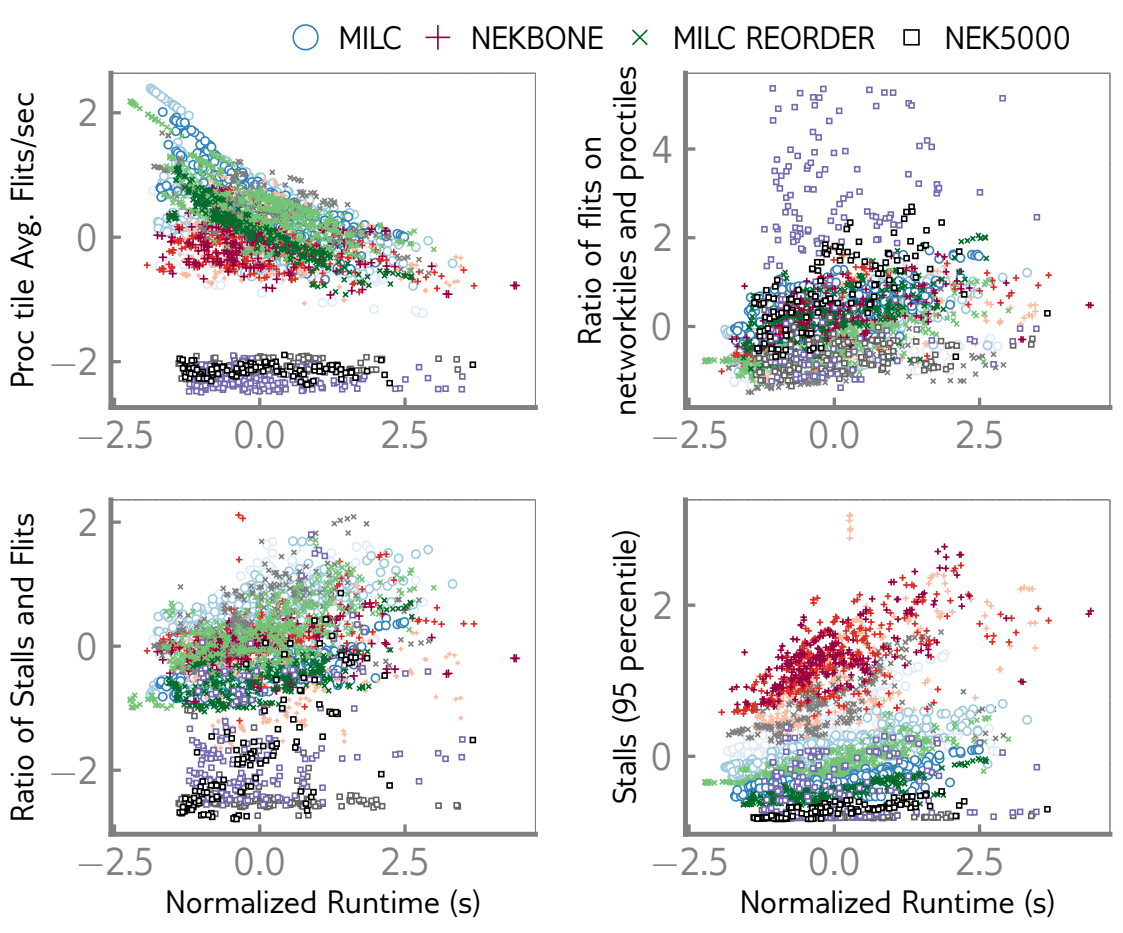
Since no single metric has the strongest correlation, hence, we use a combination of metrics

Commonalities among Applications

Shows the distribution of normalized stall-to-flit ratio of the network tiles for MILC and Nekbone
3 different node sizes, different placements; each line represent a unique run
Similar distributions, **Long tails** represent congestion, they present at the **same scale**



Commonalities among Applications after Standardization



- Top 4 features of importance
- Shows the relation between the **standardized metrics** and the **normalized runtime**
- 3 different node sizes
  - 128 nodes - light color
  - 256 nodes - medium color
  - 512 nodes - dark color

- Nek5000 shows distinct communication characteristics
- Overall, display **significant overlap** between the applications
- Hence, a **generic model** trained on runtime + Contexts for several applications to estimate variability given the runtime + Context as input is feasible

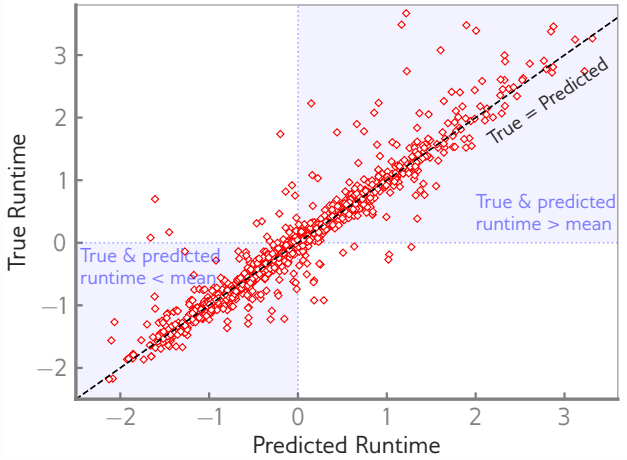
A Generic Predictive Model

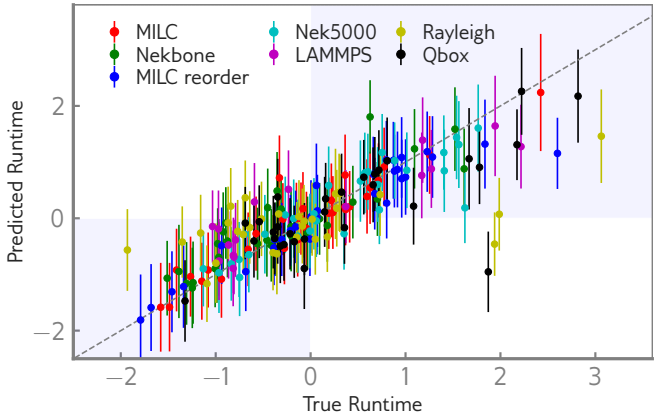
- We have experimented with various regression models such as RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor, and Support Vector Regression
- SVR performs the best
- Hyperparameter tuning is performed using gridSearchCV
- R^2 coefficient of determination metric as a scoring function

- SVR model is built using data from seven applications
- Prediction accuracy of R^2=0.91 (10-fold cross validation test)

Robustness Improvements:

- We use an ensemble model with SVR and Random Forest Regressor
- SVR reports only point estimate
- RF also reports estimation intervals along with point estimates
- Only if both Regression models "Vote" positively, the model reports the *Variability Estimate*
- *Variability Estimate* = (SVR prediction +/- the error bounds of RF)





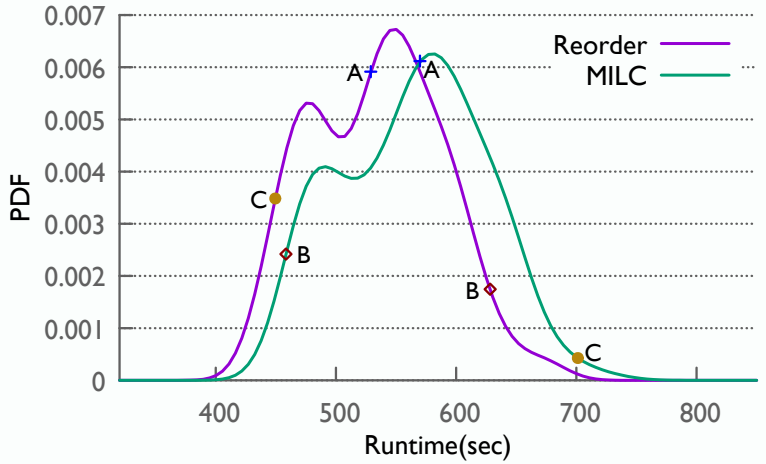
Model Use Cases

- Three runs of MILC and MILC_reorder
- Just with the runtimes, the tuning benefit is unclear
- It is not clear if a specific runtime can be used as a representative for average expected runtime

RUN	MILC		REORDER	
	Runtime	Variability Estimate	Runtime	Variability Estimate
A	569.05	-0.05 (+/- 0.58)	529.73	-0.07 (+/- 0.42)
B	458.13	-1.56 (+/- 0.76)	626.86	1.26 (+/- 0.65)
C	703.09	2.54 (+/- 0.90)	449.33	-1.50 (+/- 0.49)

- Given runtime + Context, model provides the *Variability Estimate*
- MILC Run A: *Suitable as a benchmark*
- MILC Run B: *Ran in very favorable network conditions*
- MILC Run C: *Ran under heavy congestion*

- MILC Run B and Reorder Run B *should not be compared*
- MILC Run C and Reorder Run C *also should not be compared*
- MILC Run A and Reorder Run A *are fair to compare*



The Variability Estimates + runtimes can differentiate between the tuning benefits and the congestion effects on application performance

Summary

- Network noise on latest HPC networks can cause high variability from run to run
- The application runtime by itself is insufficient to assess
  - Scaling efficiency
  - Performance tuning benefit
  - Instrumentation overhead from performance tools
- A Context that estimate the impact of network congestion on that specific run is essential
- Network counters-based metrics are used to capture the Context
- An Ensemble model using SVM and RF trained on seven production applications is constructed
  - Prediction quality of 91% on the 10-fold cross validation test
- Future work
  - Test the methodology on other current and future networks
  - Improve the model prediction quality on applications not featuring in the training set

https://xgitlab.cels.anl.gov/variability/pmbs_sc19


QUESTIONS?