# Rethinking How We Manage Failures

**Devesh Tiwari**
Goodwill Computing Lab
https://web.northeastern.edu/tiwari

Northeastern University

With sincere thanks to students and collaborators at Oak Ridge National Lab, Lawrence Berkeley National Lab, Argonne National Lab, Northeastern University, College of William & Mary and Wayne State University
... and sponsors

# There are two worlds in this world!

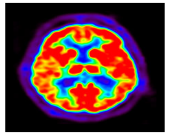
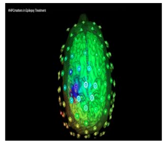



High Performance Computing
Data Centers


Enterprise Computing
Data Centers

# There are two worlds in this world!

Large-scale Computational Science Applications



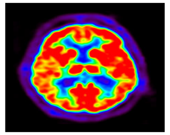
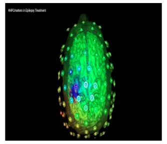
## High Performance Computing Data Centers

Latency-sensitive applications plus batch jobs


## Enterprise Computing Data Centers

# There are two worlds in this world!

Large-scale Computational Science Applications

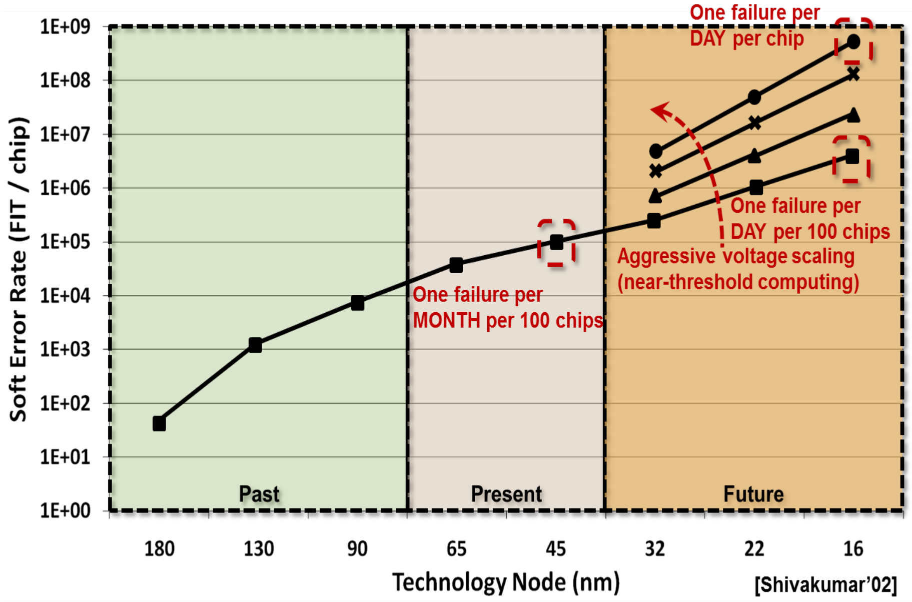



Long running "tightly-coupled" applications, explicit focus on resilience mechanisms (e.g., checkpoint-restart), improvements in system throughput and utilization desired.

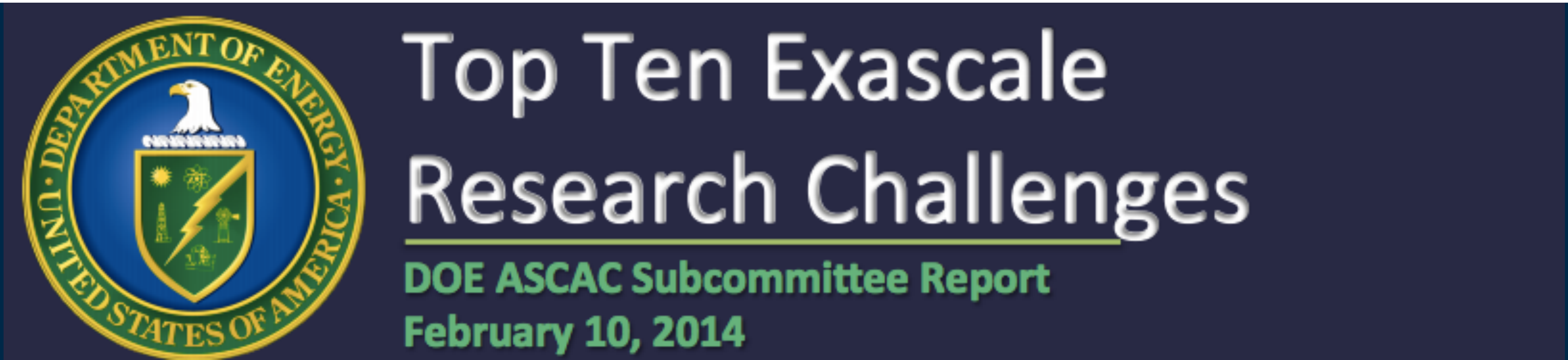
Latency-sensitive applications plus batch jobs

MEMCACHED NGINX Spark GraphLab mongoDB Google hadoop


Short running jobs (in the order of milliseconds to seconds), restart-on-failure, focus on achieving tighter SLAs and reducing tail latency


# Large-scale scientific applications will face severe resilience challenge at exascale!



# Rethinking How We Manage Failures

*You can't avoid them, you can't predict them,*
*but you can choose who gets hit by them!*

**Key is to exploit statistical properties of failures and diversity in characteristics of jobs**

# Who gets hit by failures: Part I

# Who gets hit by failures in time!

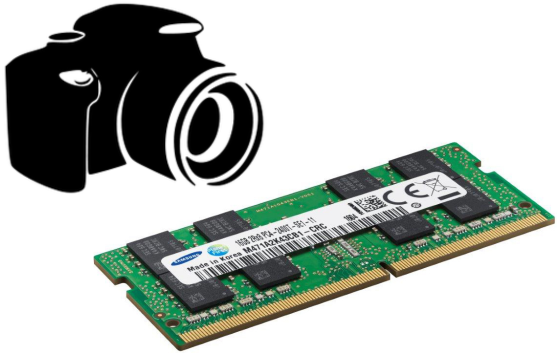
Garg et al., "Shiraz: Exploiting System Reliability and Application Resilience Characteristics to Improve Large Scale System Throughput", DSN 2018.

# What new territory does this work explore?


**Prior efforts increase effective system MTBF**

Building more reliable system components

Failure prediction, Quarantine job scheduling



**Prior efforts reduce checkpointing overhead**

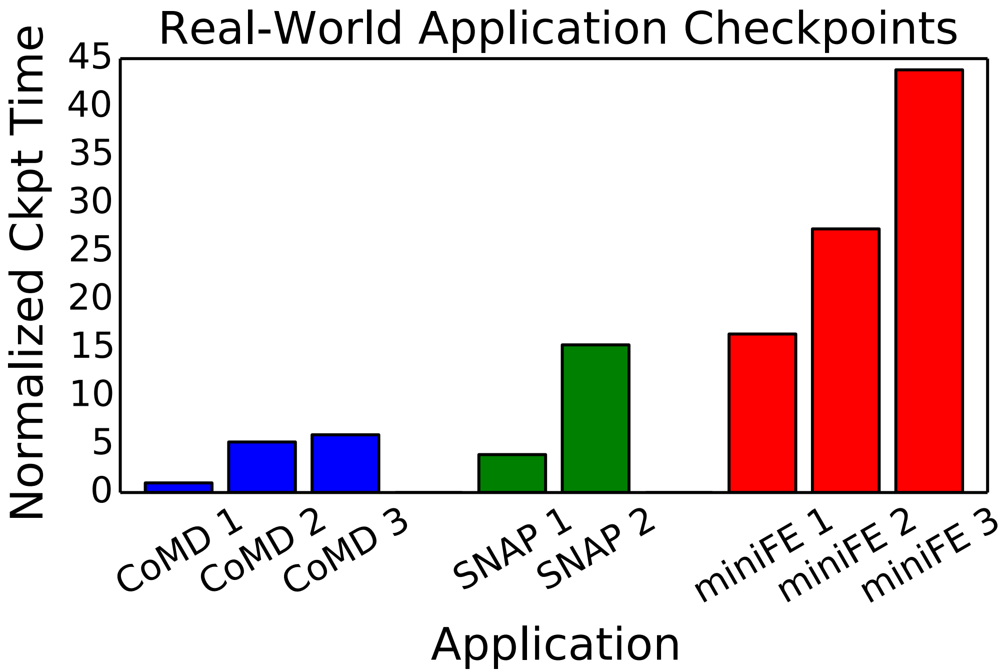
Incremental checkpointing of system state

Checkpoint compression, Lazy checkpointing

**Shiraz improves both system throughput and individual application performance by exploiting (a) <u>differences in application resilience characteristics,</u> and (b) <u>dynamic system reliability behavior</u>**

# Observation 1: Large variations exist in checkpointing overheads

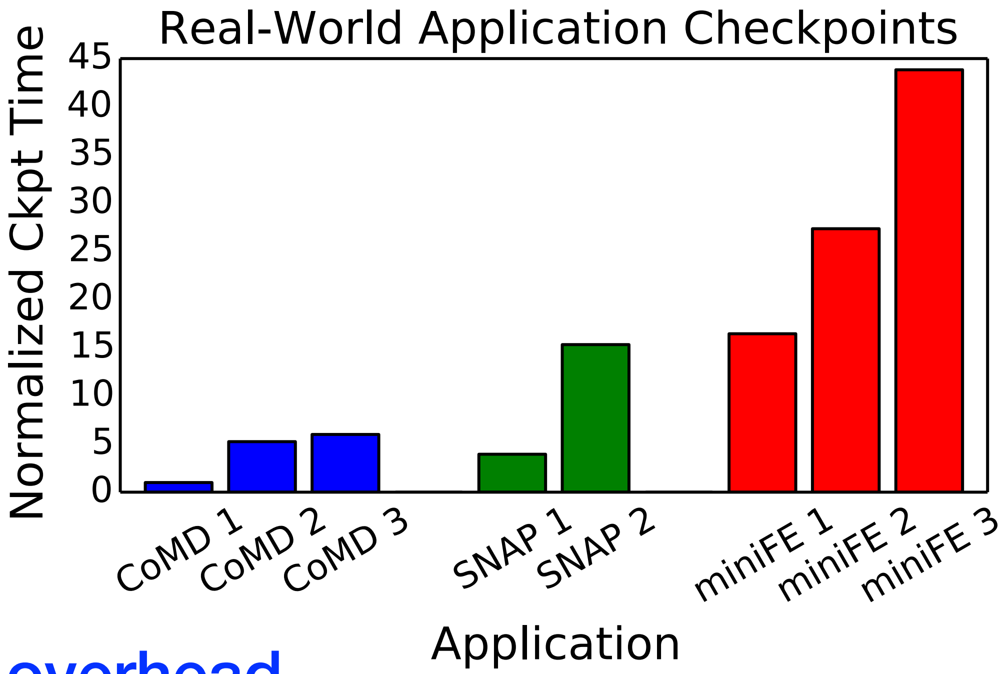
| App. Domain | Checkpoint time |
|---|---|
| Climate | 2 seconds |
| Molecular Simulations | 6 seconds |
| Combo-crunch | 50 seconds |
| Laser Plasma | 30 minutes |
| Plasma Science | 45 minutes |




**Checkpointing overhead varies with application type, simulation parameters, memory-resident data size, input size, etc.**

# Observation 1: Large variations exist in checkpointing overheads

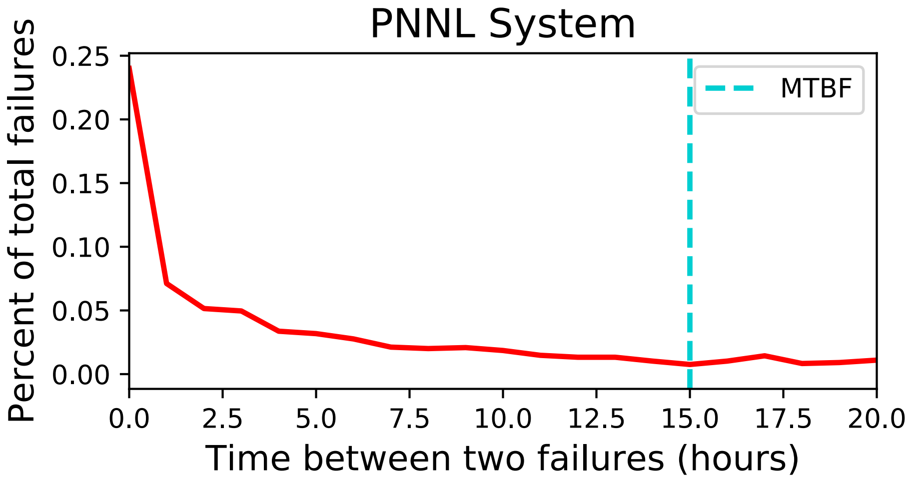
| Machine | Application Domain | Checkpointing Duration (sec.) |
|---|---|---|
| Titan (OLCF) | Climate Change Simulation with the Community Earth System Model | 1.5 |
| Hopper (NERSC) Franklin (NERSC) | 20th Century Reanalysis | 2 |
| Jaguar (ORNL) Hopper (NERSC) | Molecular Simulation in Energy Biosciences | 6 |
| Carver and Euclid (NERSC) | Computational Predictions of Trans. Factor Binding Sites | 50 |
| Cori (NERSC) | Chombo-crunch | 70 |
| Hopper (NERSC) | Climate Science for a Sustainable Energy Future | 150 |
| Hopper (NERSC) | Laser Plasma Interactions | 1800 |
| Hopper (NERSC) | Plasma Based Accelerators | 2000 |
| Hopper (NERSC) | Plasma Science Studies | 2700 |

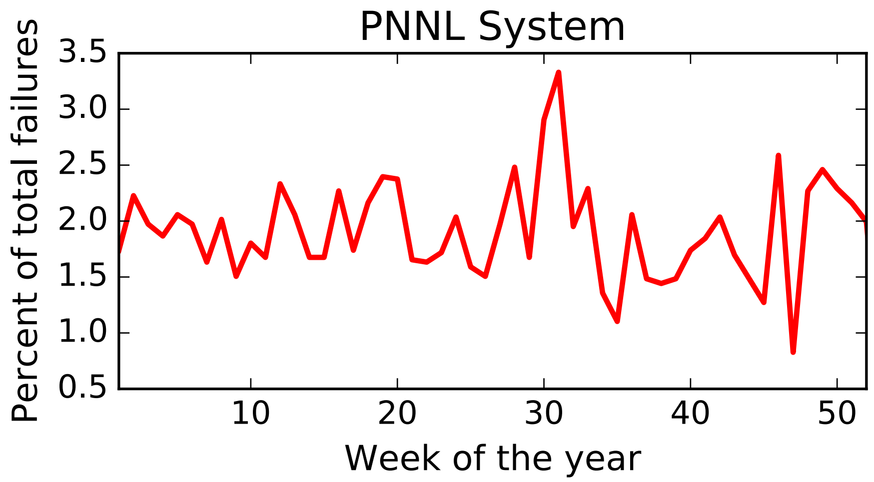



Light Applications (LW): low checkpointing overhead lower optimal checkpointing interval: $\mathrm{OCI}_{\mathrm{LW}} = \sqrt{2M\delta_{\mathrm{LW}}}$

Heavy Applications (HW): high checkpointing overhead; higher optimal checkpointing interval: $\mathrm{OCI}_{\mathrm{HW}} = \sqrt{2M\delta_{\mathrm{HW}}}$

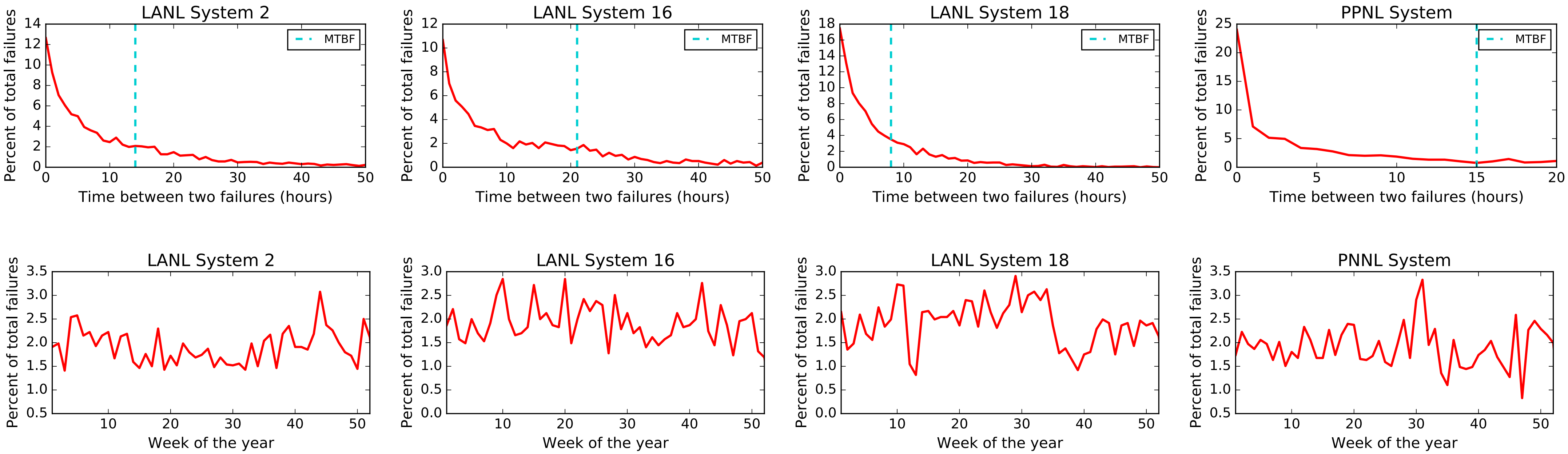
# Observation 2: System failure rate is not constant over time




**The hazard rate monotonically decreases between two failures, although that does not imply that the system becomes more or less reliable over a long period of time**

# Observation 2: System failure rate is not constant over time

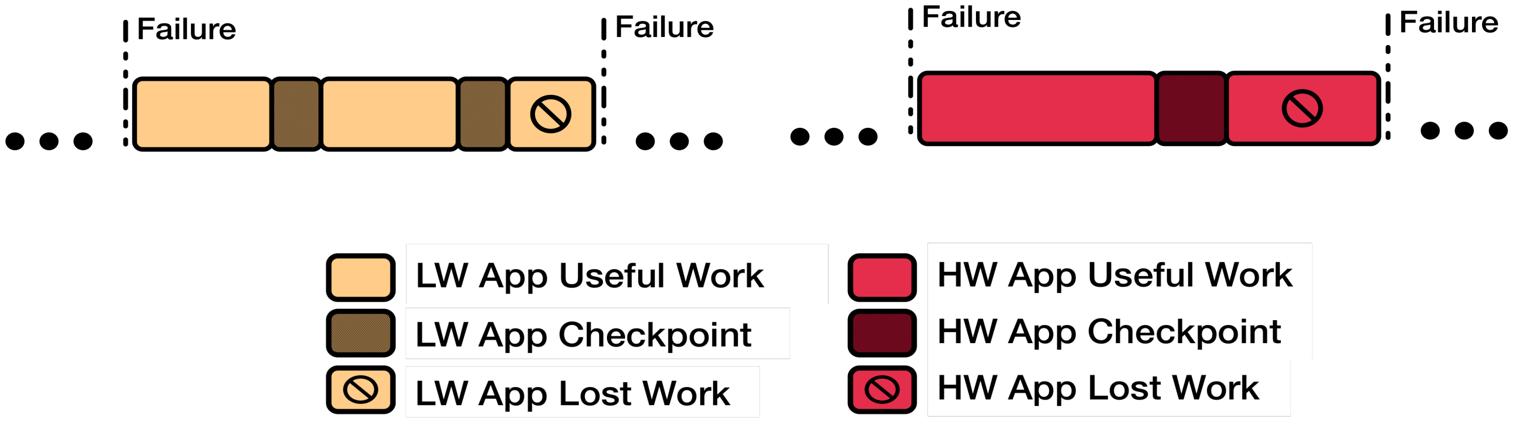



**The hazard rate monotonically decreases between two failures, although that does not imply that the system becomes more or less reliable over a long period of time**

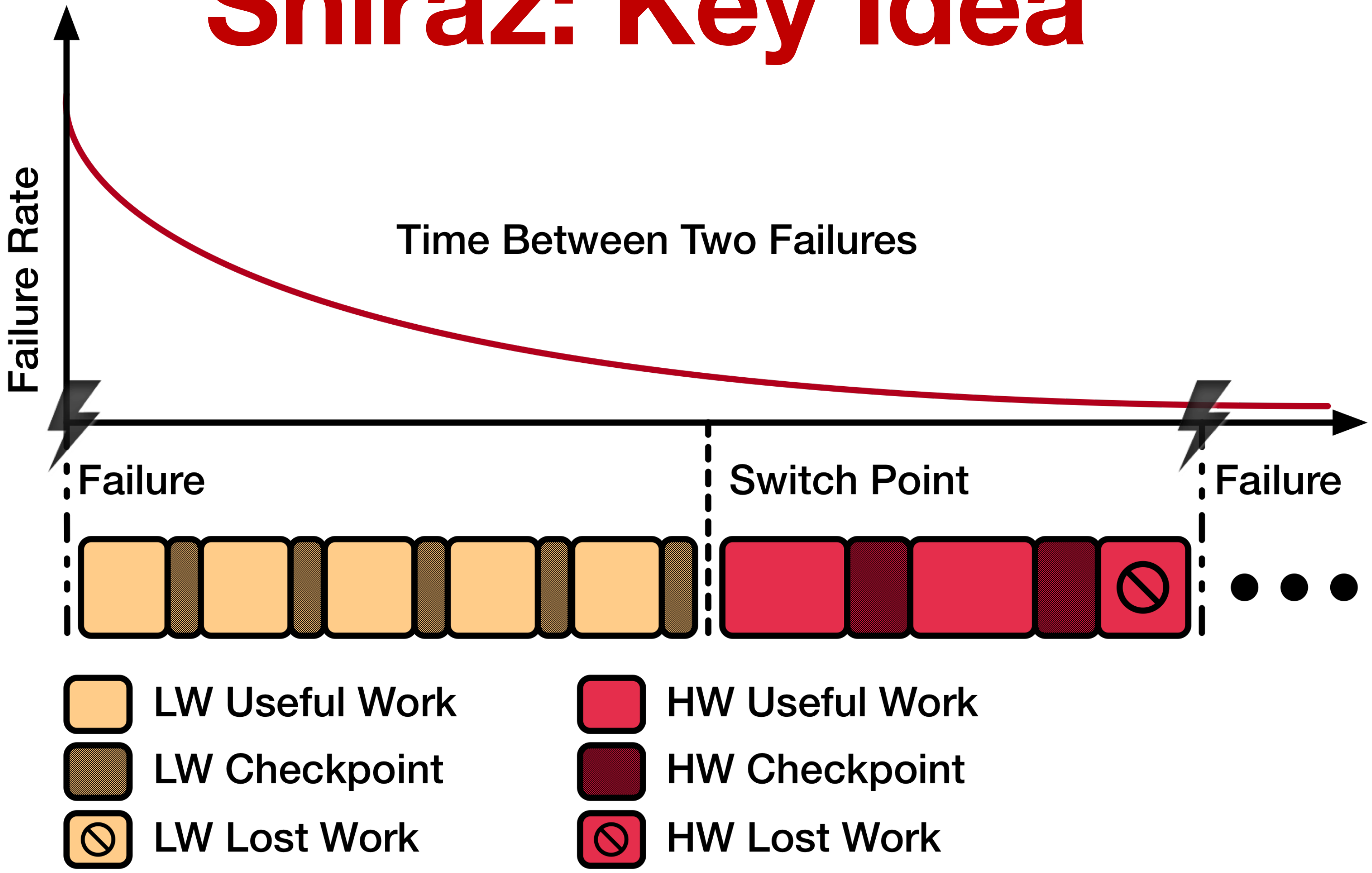
# Observation III: Conventional scheduling is ineffficient

Switch between applications at every failure


**Light applications (LW) have lower average lost work per failure compared to heavy applications (HW)**



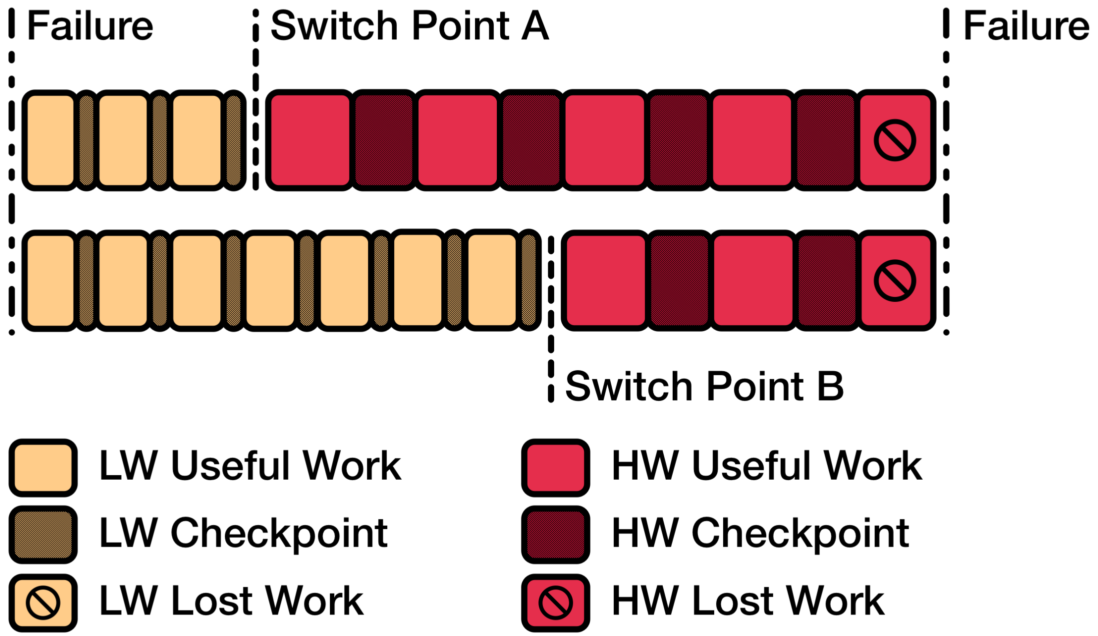
# Shiraz: Key Idea

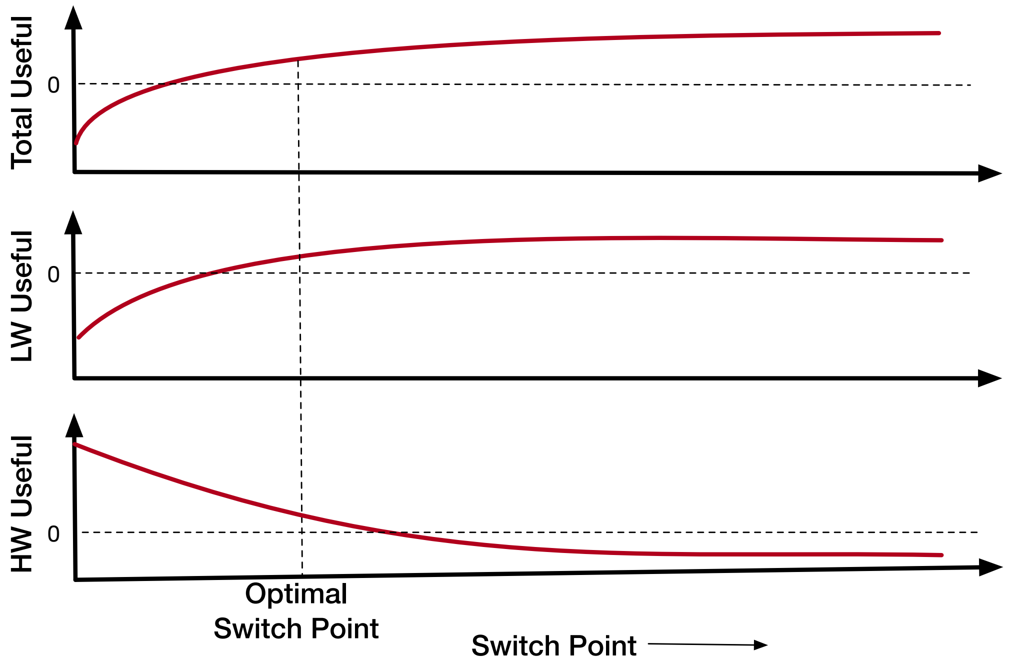



Schedule a light-weight application after a failure, and switch to a heavy-weight application, at an *optimal point* between two failures

# Optimal Switching Point

Optimal switching point is the number of checkpoints that LW application takes before yielding to the HW application such that the system throughput (i.e., total useful work per failure) is maximized, without hurting any application's performance



# Shiraz Model

- **Inputs** MTBF (M), Checkpointing overheads ($\delta_{LW}$ and $\delta_{HW}$)

  $\delta$-factor = $\delta_{HW}$ / $\delta_{LW}$ (ratio of checkpointing overheads)

- **Output** Optimal switching point (k) -- the number of checkpoints by LW application before scheduling HW application

$$
\begin{aligned}
& T^{\text{LW}}_{\text{useful-shiraz}} - T^{\text{LW}}_{\text{useful-base}} = T^{\text{HW}}_{\text{useful-shiraz}} - T^{\text{HW}}_{\text{useful-base}} \\
\text{s.t.} \quad & (T^{\text{LW}}_{\text{useful-shiraz}} - T^{\text{LW}}_{\text{useful-base}}) \geq 0 \\
\text{and} \quad & (T^{\text{HW}}_{\text{useful-shiraz}} - T^{\text{HW}}_{\text{useful-base}}) \geq 0
\end{aligned}
$$

More modeling details and tricks in the paper

$$\text{Fail}^{\text{num}}_{(t_{\text{start}}, t_{\text{end}})} = \frac{T_{\text{total}}}{M} \times (e^{-(\frac{t_{\text{start}}}{\lambda})^{\beta}} - e^{-(\frac{t_{\text{end}}}{\lambda})^{\beta}})$$

$$T^{\text{LW}}_{\text{lost-base}} = \epsilon \times (\text{OCI}_{\text{LW}} + \delta_{\text{LW}}) \times \text{Fail}^{\text{num}}_{\text{total}}$$

$$T^{\text{HW}}_{\text{lost-base}} = \epsilon \times (\text{OCI}_{\text{HW}} + \delta_{\text{HW}}) \times \text{Fail}^{\text{num}}_{\text{total}}$$

$$T^{\text{LW}}_{\text{useful-base}} = \sum_{i=1}^{\infty} i \times \text{OCI}_{\text{LW}} \times \text{Fail}^{\text{num}}_{i,i+1}(\text{OCI}_{\text{LW}} + \delta_{\text{LW}})$$

$$T^{\text{HW}}_{\text{useful-base}} = \sum_{i=1}^{\infty} i \times \text{OCI}_{\text{HW}} \times \text{Fail}^{\text{num}}_{i,i+1}(\text{OCI}_{\text{HW}} + \delta_{\text{HW}})$$

$$T^{\text{LW}}_{\text{useful-shiraz}} = \sum_{i=1}^{k} i \times \text{OCI}_{\text{LW}} \times \text{Fail}^{\text{num}}_{i,i+1}(\text{OCI}_{\text{LW}} + \delta_{\text{LW}})$$

$$T^{\text{HW}}_{\text{useful-shiraz}} = \sum_{i=k}^{\infty} i \times \text{OCI}_{\text{HW}} \times \text{Fail}^{\text{num}}_{i,i+1}(\text{OCI}_{\text{HW}} + \delta_{\text{HW}})$$

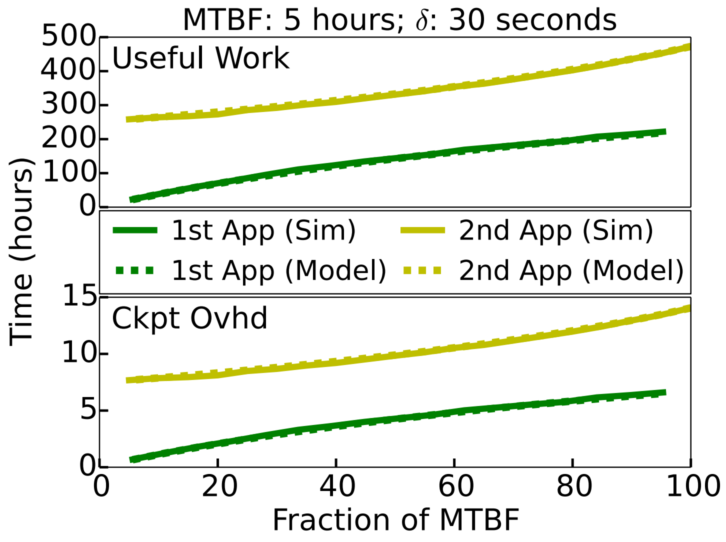
# Shiraz Model Validation

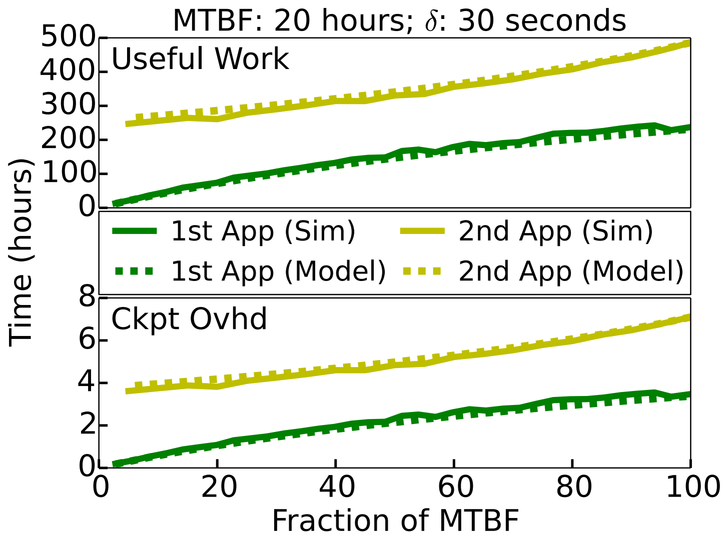
Shiraz's per-application predictions validated against simulation

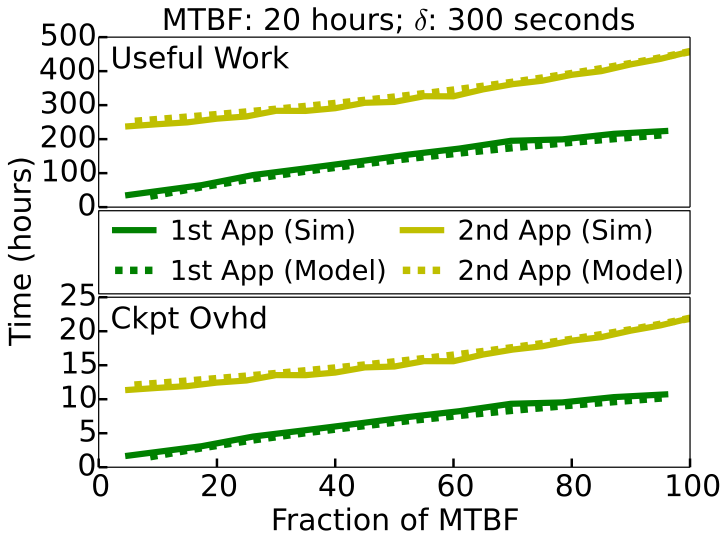
- Useful work, checkpointing overhead, and lost work
- Different system scale and storage system I/O bandwidth
- Optimal switching point

**Extensive validation in the paper**

No assumptions about order and type of scheduled applications



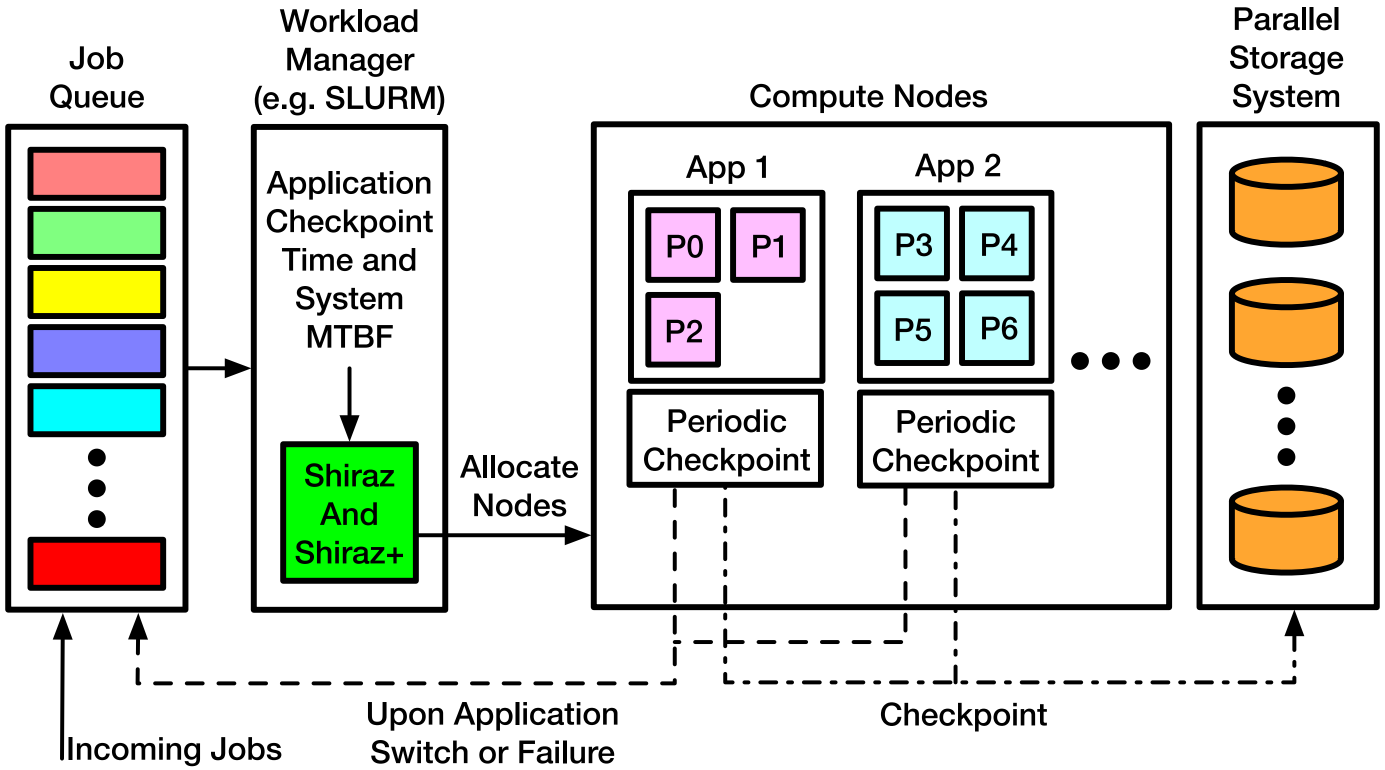



# Shiraz Optimal Switching Points

| System Type | Checkpointing Overhead Ratio | Model Optimal Switching Point | Simulation Optimal Switch Point |
|---|---|---|---|
| Exascale | 5x | 6 | 6 |
| Exascale | 25x | 13 | 13 |
| Exascale | 100x | 26 | 26 |
| Exascale | 1000x | 81 | 79 |
| Petascale | 5x | 12 | 11 |
| Petascale | 25x | 26 | 24 |
| Petascale | 100x | 51 | 51 |
| Petascale | 1000x | 161 | 161 |

**Shiraz model accurately predicts the optimal switch point across different scales and different checkpointing overhead ratios**

# Shiraz Evaluation

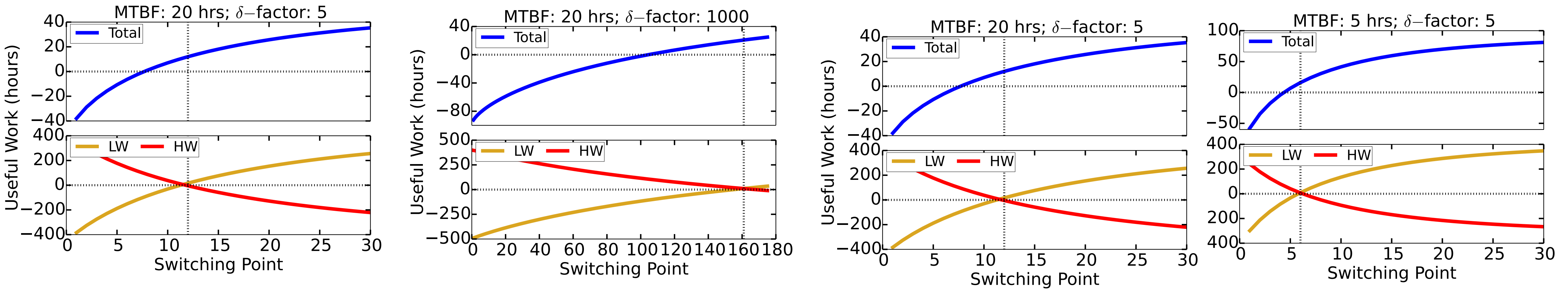



Exploration of real-world system parameters through simulations

Peta/Exa- scale, varying checkpointing overheads, multiple applications

Real-world prototype on a cluster with system-level checkpointing

# Optimal Switching Point: Insights




**Checkpoint overhead ratio increases**

**Petascale to Exascale System MTBF decreases**

**Optimal switching point shifts to the right and benefits increase as the difference in the time-to-checkpoint between applications increases**

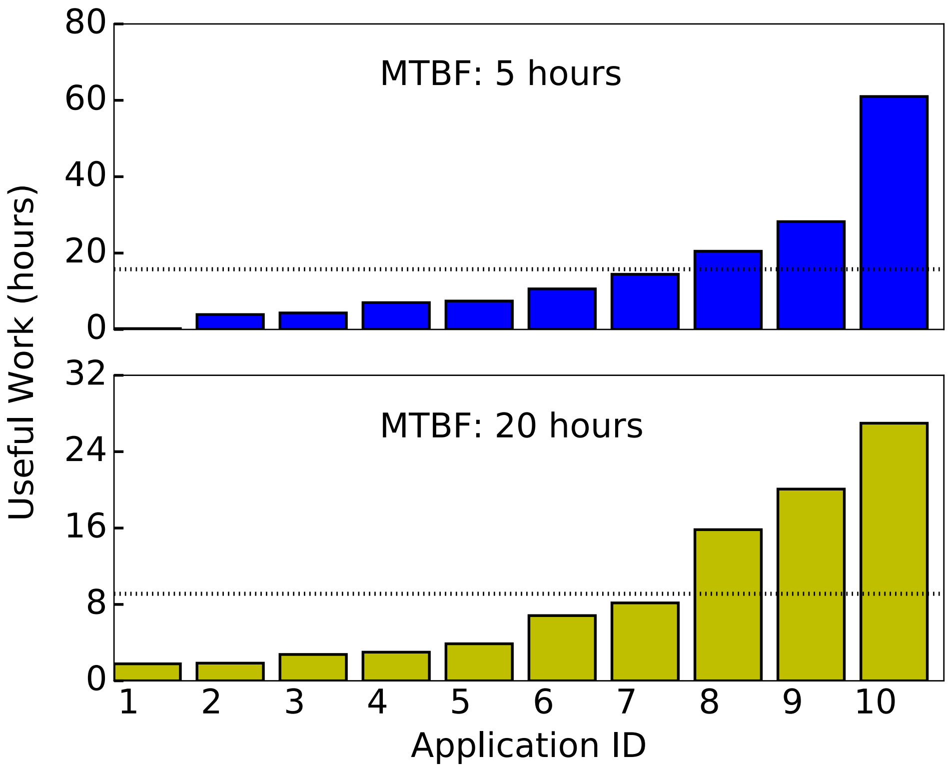
**Optimal switching point shifts to the left as the MTBF decreases but the benefits increase (i.e., Shiraz is more useful at exascale)**

# Optimal Switching Point: Insights

Intuitively, one may think the optimal switching point to be half of the MTBF (in terms of time), but Shiraz discovers that it can be larger than even the MTBF in many cases!

# Shiraz is Effective in Multi-Job Mix

| Machine | Application Domain | Checkpointing Duration (sec.) |
|---|---|---|
| Titan (OLCF) | Climate Change Simulation with the Community Earth System Model | 1.5 |
| Hopper (NERSC) Franklin (NERSC) | 20th Century Reanalysis | 2 |
| Jaguar (ORNL) Hopper (NERSC) | Molecular Simulation in Energy Biosciences | 6 |
| Carver and Euclid (NERSC) | Computational Predictions of Trans. Factor Binding Sites | 50 |
| Cori (NERSC) | Chombo-crunch | 70 |
| Hopper (NERSC) | Climate Science for a Sustainable Energy Future | 150 |
| Hopper (NERSC) | Laser Plasma Interactions | 1800 |
| Hopper (NERSC) | Plasma Based Accelerators | 2000 |
| Hopper (NERSC) | Plasma Science Studies | 2700 |




**Shiraz extended to a multi-job mix by intelligent application pairing**

**No application is hurt in the job mix for a representative workload mix**

**Throughput improvement increases at exascale (total 157 hours)**

# Shiraz: Energy Saving Analysis

Representative workload mix 40 jobs (only 5 heavy-weight and rest 35 light-weight applications) run for a year

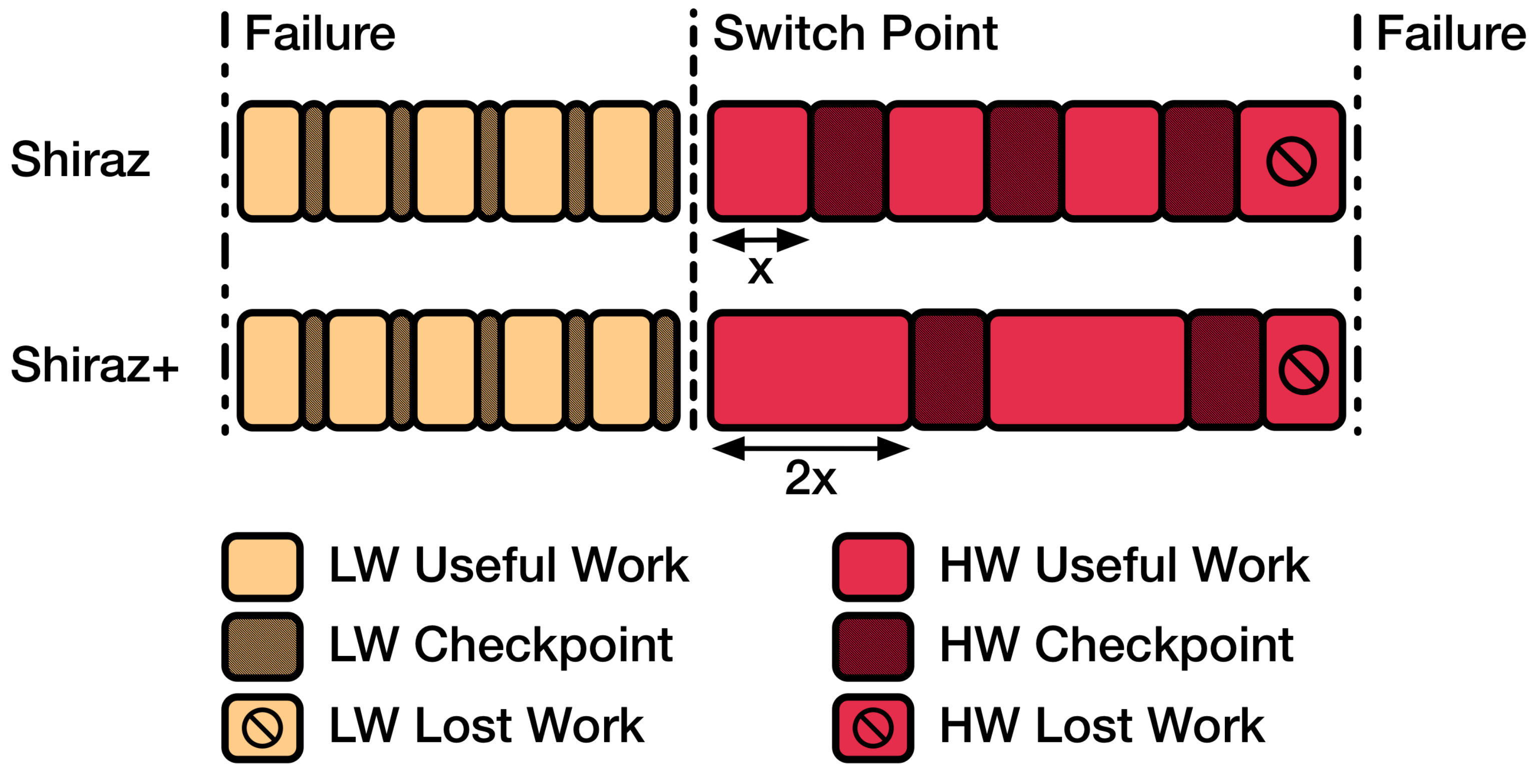
Shiraz results in savings* of $57K per annum for a 10 MW Petascale system (MTBF: 20 hours); savings of $285K over a lifetime of 5 years

Shiraz results in savings* of $178K per annum for a 20 MW future Exascale system (MTBF: 5 hours); savings for $890K over a lifetime of 5 years

* Electricity @ $0.1 per KW-hour

**Shiraz improves system throughput, but does not mitigate the I/O overhead!**

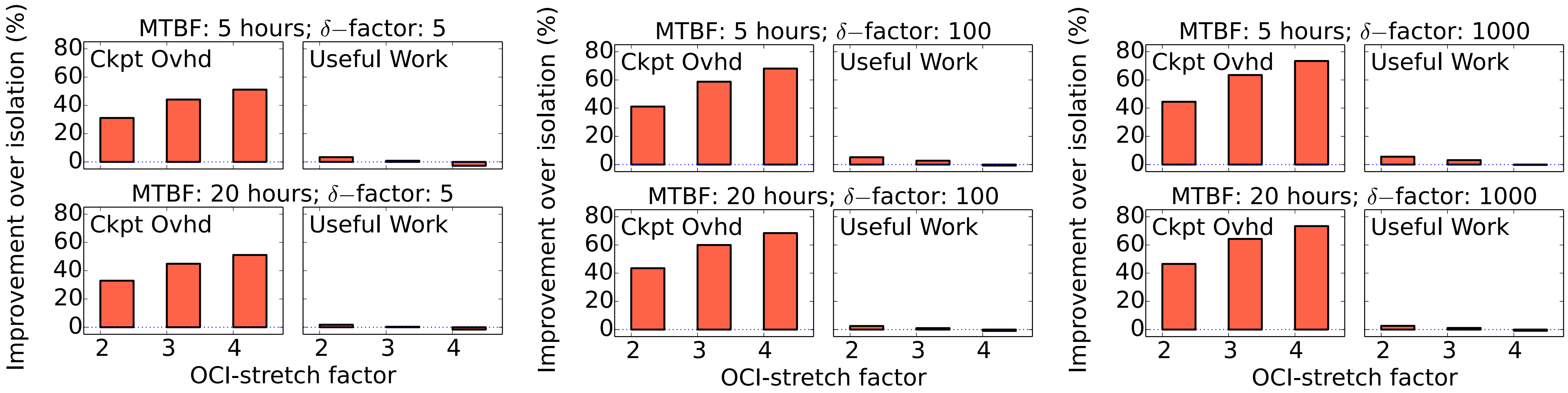
# Shiraz+: Key Idea




**Use the throughput gains obtained by Shiraz to reduce the checkpointing overhead of HW application**

Intuition: HW application is already running in a high-reliability zone

# Shiraz+ Reduces I/O Overhead



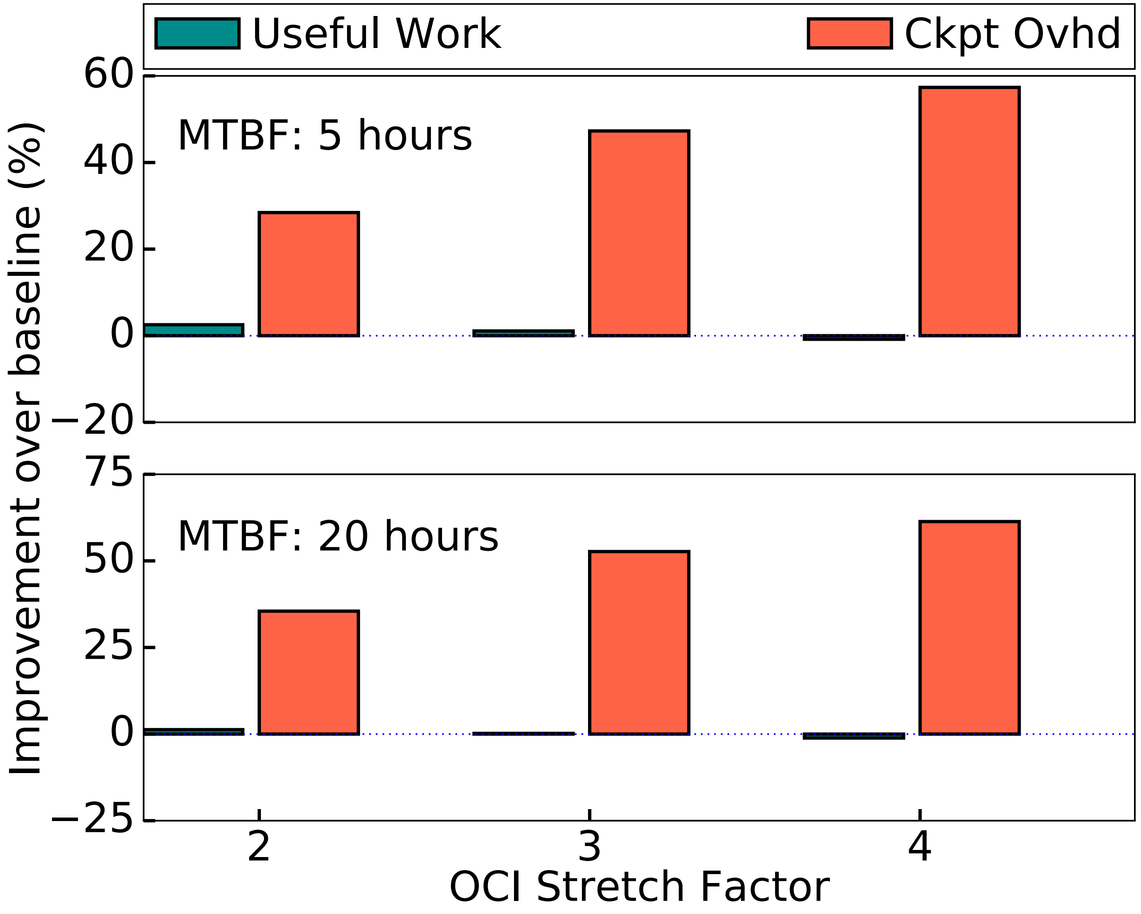

**60% reduction in checkpointing overhead when checkpointing frequency is reduced by 4x; throughput degrades only by 4.8%**

**No throughput degradation with 2x reduction in checkpointing frequency and 40% reduction in checkpointing overhead**

# Shiraz+ is Effective in Multi-Job Mix

Shiraz+ can reduce the overall checkpointing overhead by 52%, without degrading the system throughput (with 3x OCI)

With 4x OCI, the overall checkpointing overhead reduces by up to 60%, with a throughput degradation of 1%



# Who gets hit by failures: Part II

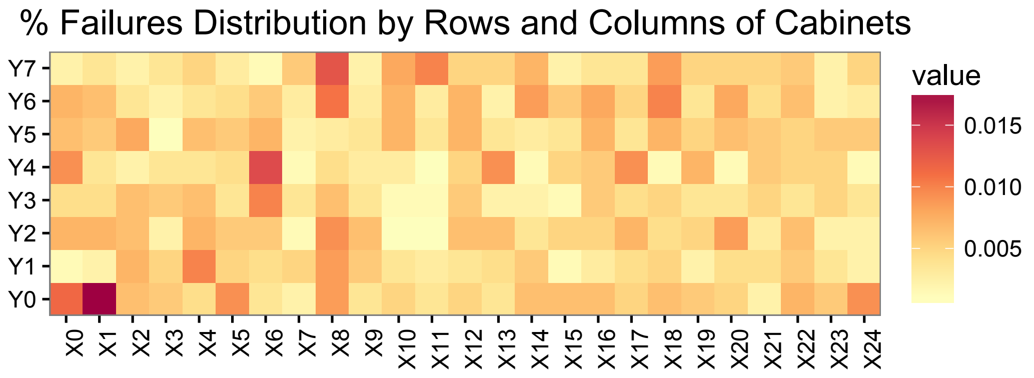
# Who gets hit by failures in space!

"Failures in Large Scale Systems: Long-term Measurement, Analysis, and Implications", SC 2017.

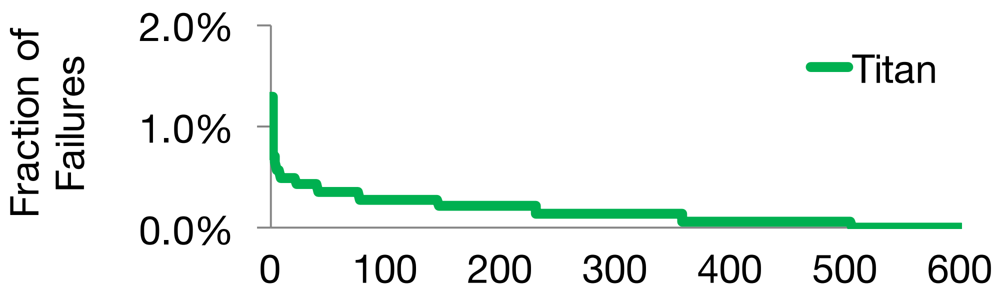
"Understanding and Exploiting Spatial Properties of System Failures on Extreme- Scale HPC Systems", DSN 2015.

# Uneven Spatial Failure Distribution

## Titan XK7

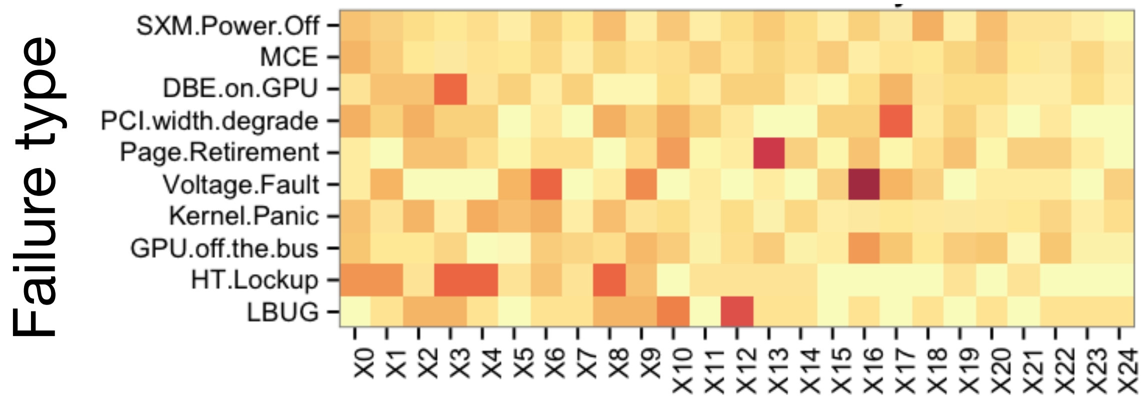



Cabinet level distribution

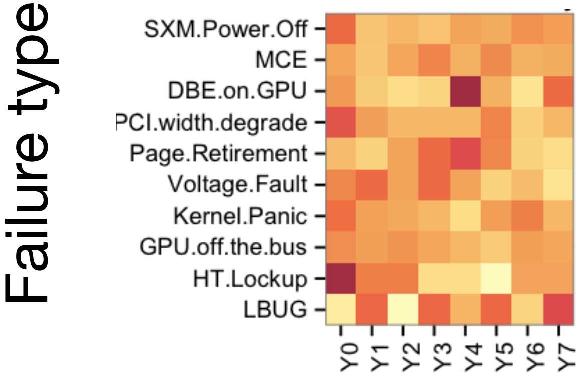



Cage level distribution

## Titan XK7



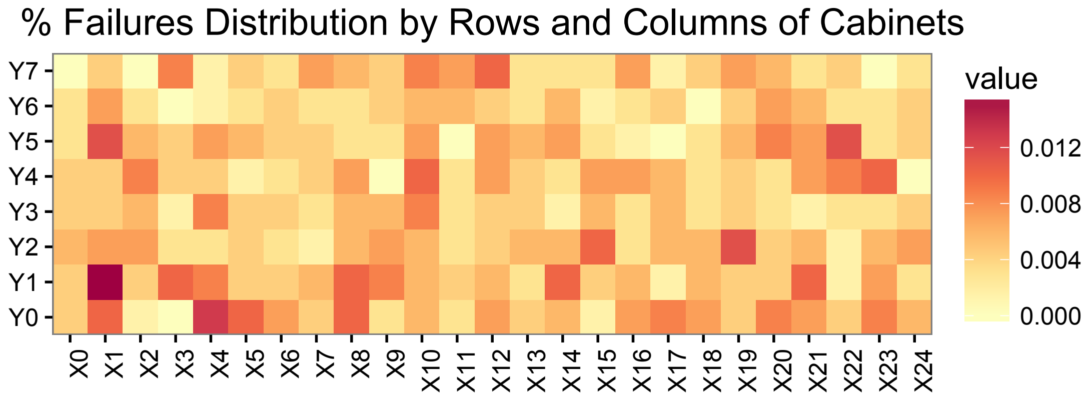

Cabinet columns



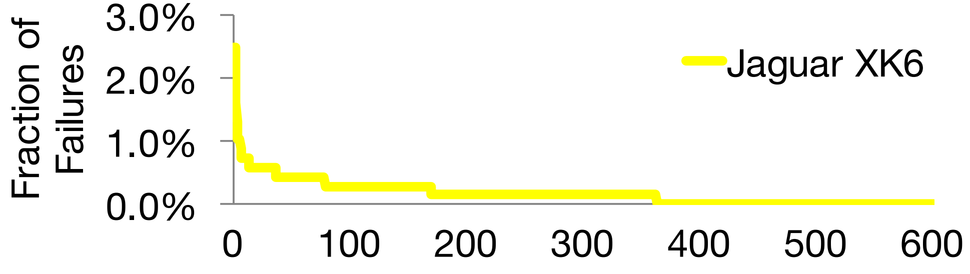

Cabinet rows

# Holds True for Other Systems too!

## Jaguar XT5

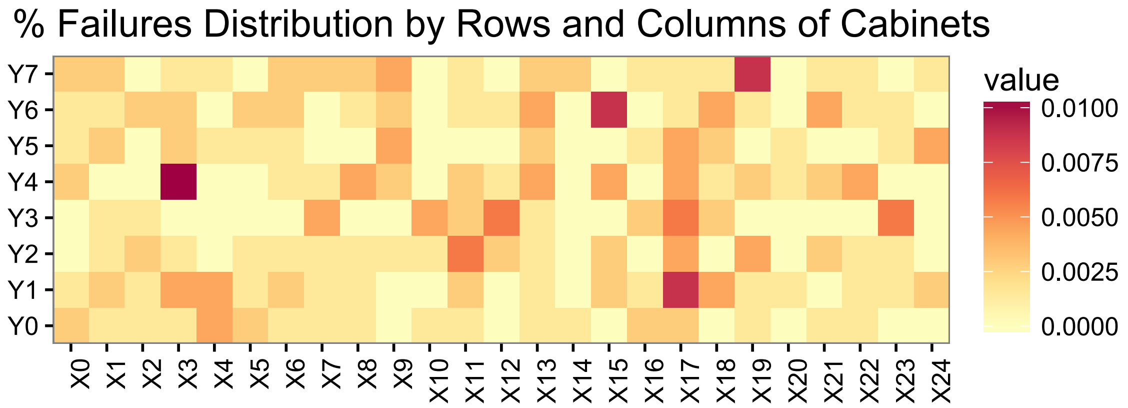



Cabinet level distribution

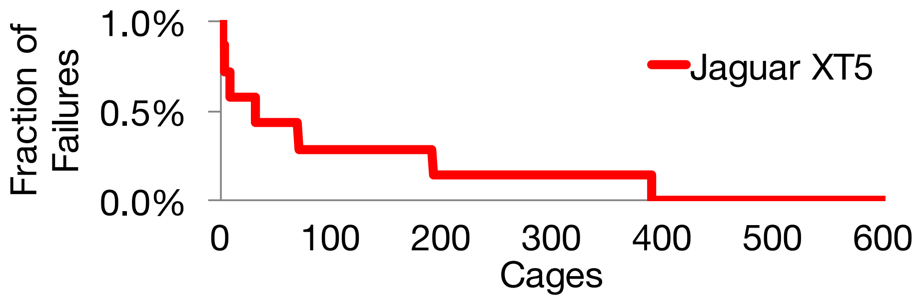



Cage level distribution

## Jaguar XK6

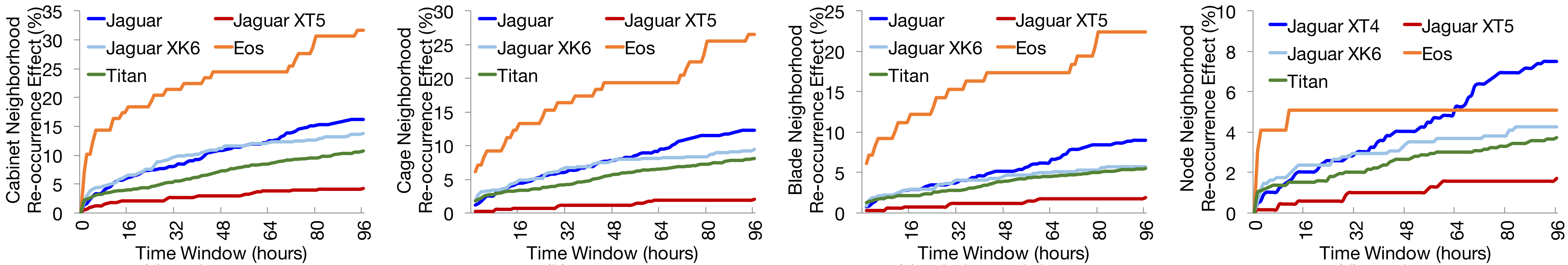



Cabinet level distribution




Cage level distribution

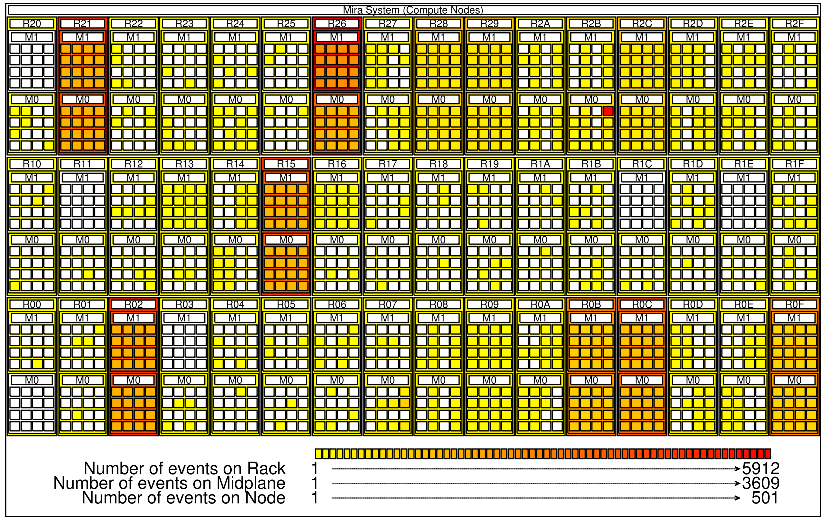
# Neighborhood Recurrence Property of System Failures



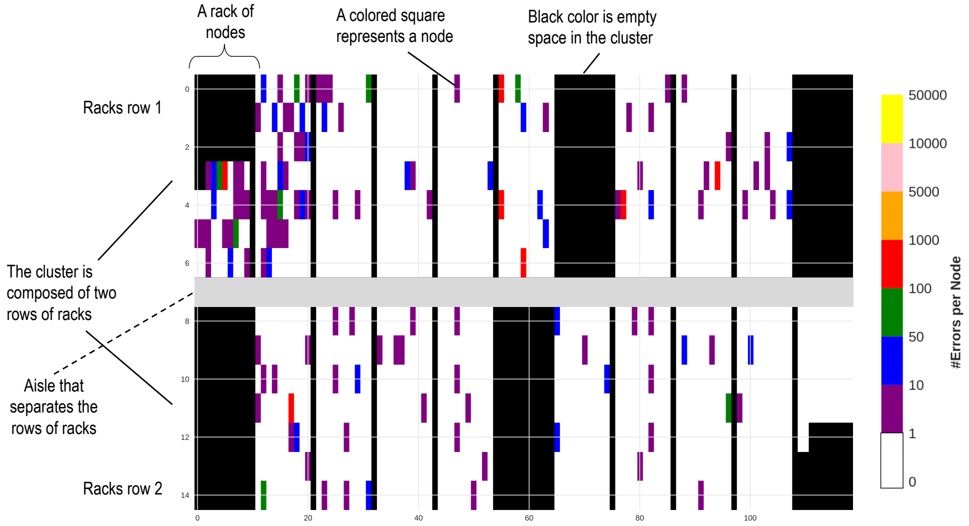

(a) Cabinet (b) Cage (c) Blade/Module (d) Node

# Evidences Supporting Neighborhood Recurrence Property by <u>Other Researchers for Other Systems</u>!

Di et al., Exploring Properties and Correlations of Fatal Events in a Large-Scale HPC System, TPDS 2019

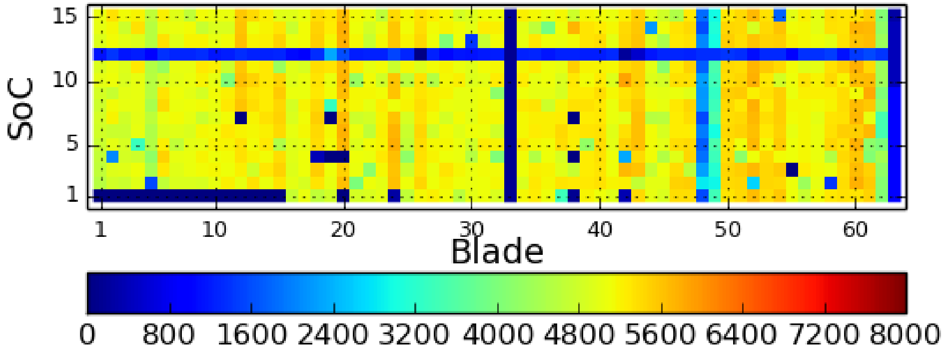



Patwari et al., Exploring Properties and Correlations of Fatal Events in a Large-Scale HPC System, FTXS 2017

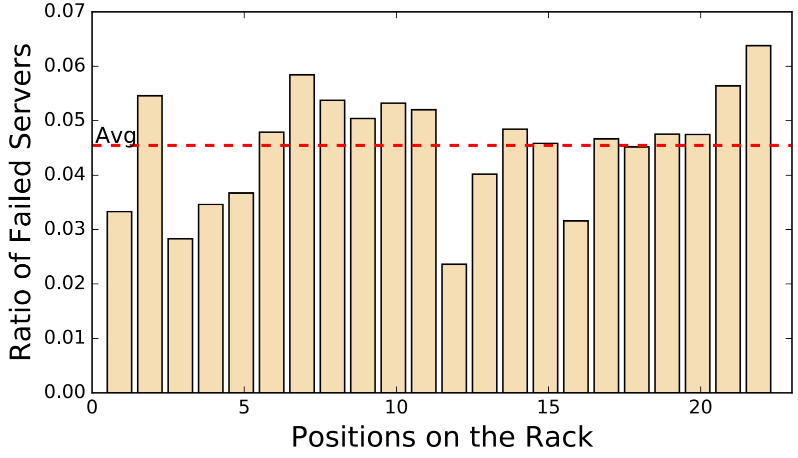


# Evidences Supporting Neighborhood Recurrence Property by Other Researchers for Other Systems!

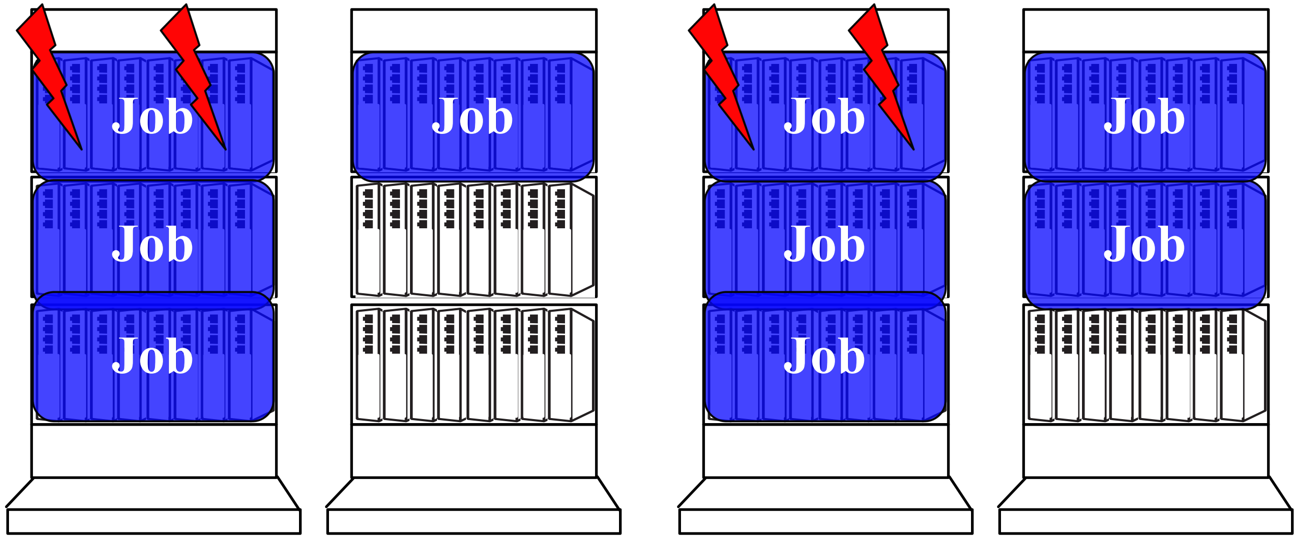
Bautista-Gomez et al., Unprotected Computing : A Large-Scale Study of DRAM Raw Error Rate on a Supercomputer, SC 2016




Wang et al., What Can We Learn from Four Years of Data Center Hardware Failures?, DSN 2017

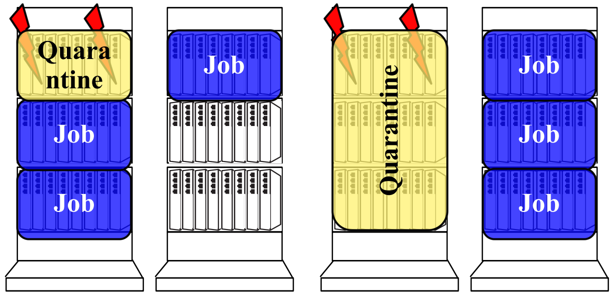


# Exploiting Spatial Locality for Improving the Effective Reliability



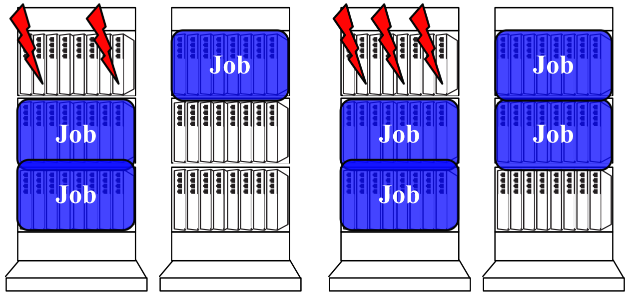

On job restart or a new job allocation
a fraction of compute capacity is not utilized or is allocated
to lower-priority / smaller jobs

# Quarantine: Design Challenges



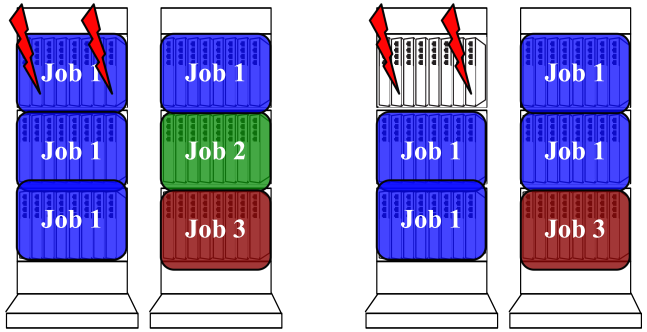

## Quarantine Granularity

Fraction of avoided system failures versus compute resource waste




## Quarantine Time Duration

Diminishing returns on the number of avoided failures

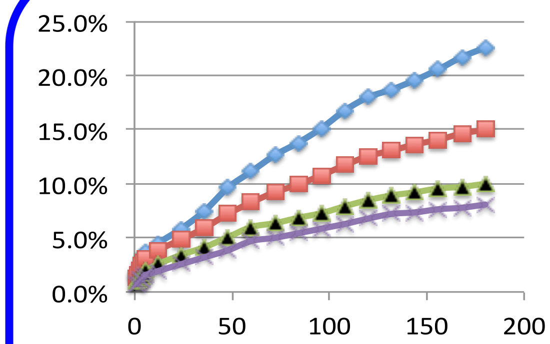



## System Utilization vs. Reliability

Trading-off lower system utilization for improved reliability

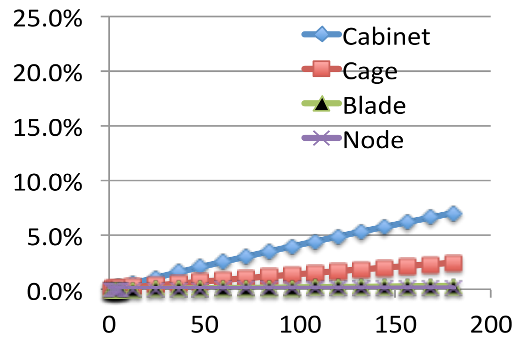
# Quarantine Technique: In Action


## System Reliability
## Fraction of failures avoided

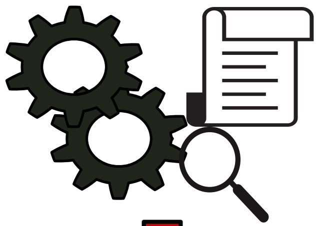



Quarantine hours

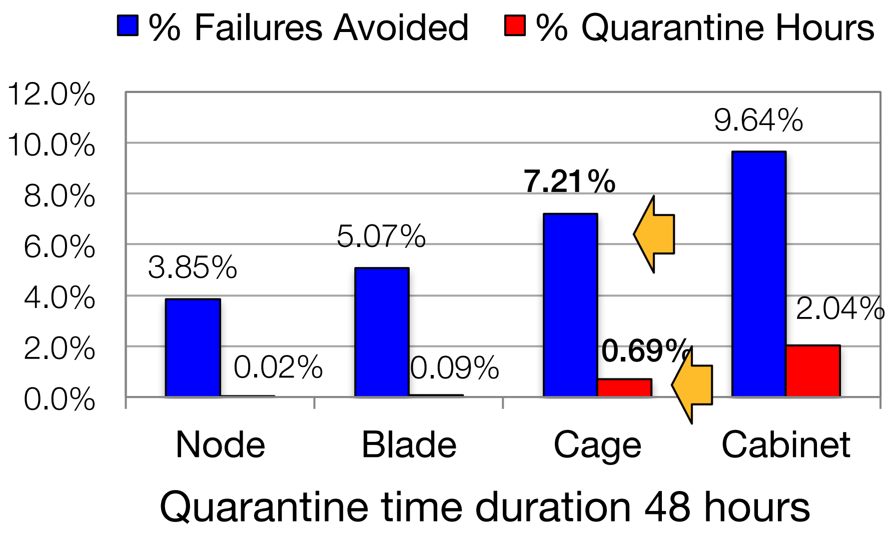
## System Utilization
## Quarantine node hours

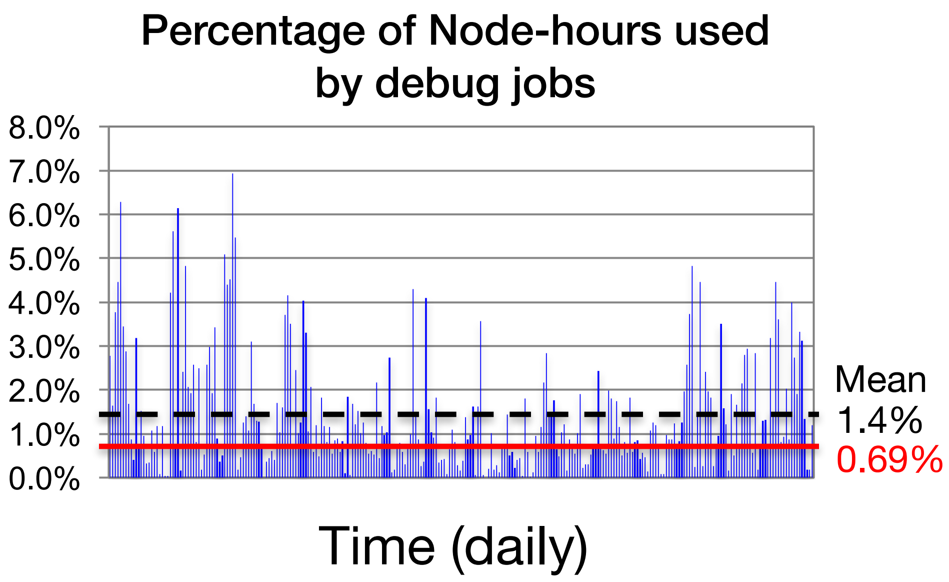



Quarantine hours



**Feedback to the job scheduler**

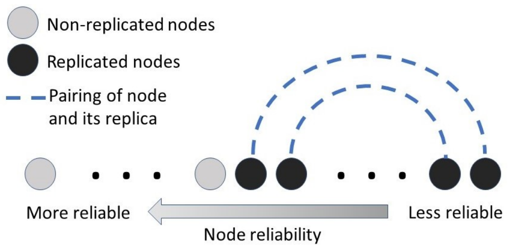


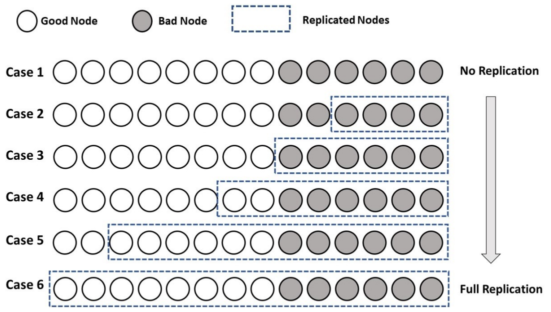




**A significant fraction of failures can be avoided from interrupting production or critical applications and scheduling debug jobs in the quarantine region**

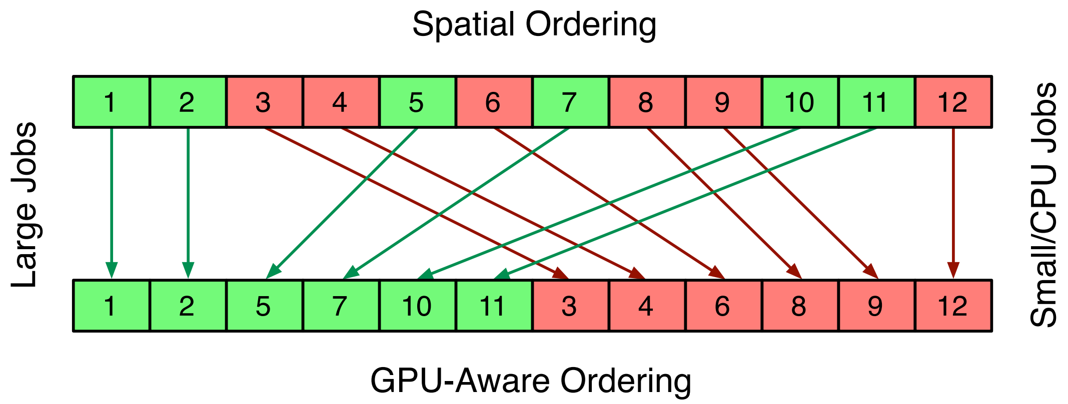
# Interesting Use Cases

Hussian et al., Partial Redundancy in HPC Systems with Non-Uniform Node Reliabilities, SC 2018




Zimmer et al., GPU age-aware scheduling to improve the reliability of leadership jobs on Titan, SC 2018



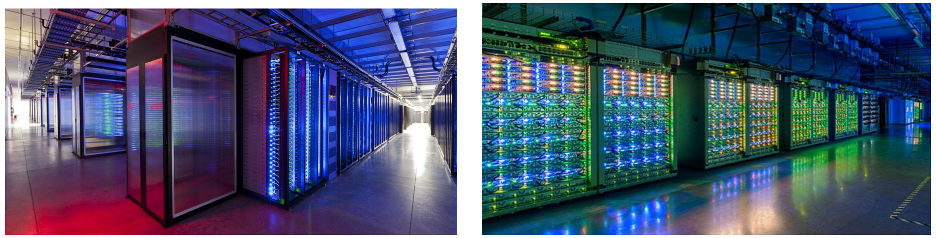
# There are two worlds in this world!

Large-scale Computational Science Applications


High Performance Computing
Data Centers

Latency-sensitive applications plus batch jobs

MEMCACHED NGINX mongoDB Google Spark GraphLab hadoop



Enterprise Computing
Data Centers

# Vibration Effects of Storage Devices

"What does Vibration do to Your SSD?" Janki Bhimani, Tirthak Patel, Ningfang Mi, Devesh Tiwari, In the Proceedings of the 56th Annual Design Automation Conference (DAC), 2019.

# We know vibration hurts hard disks!

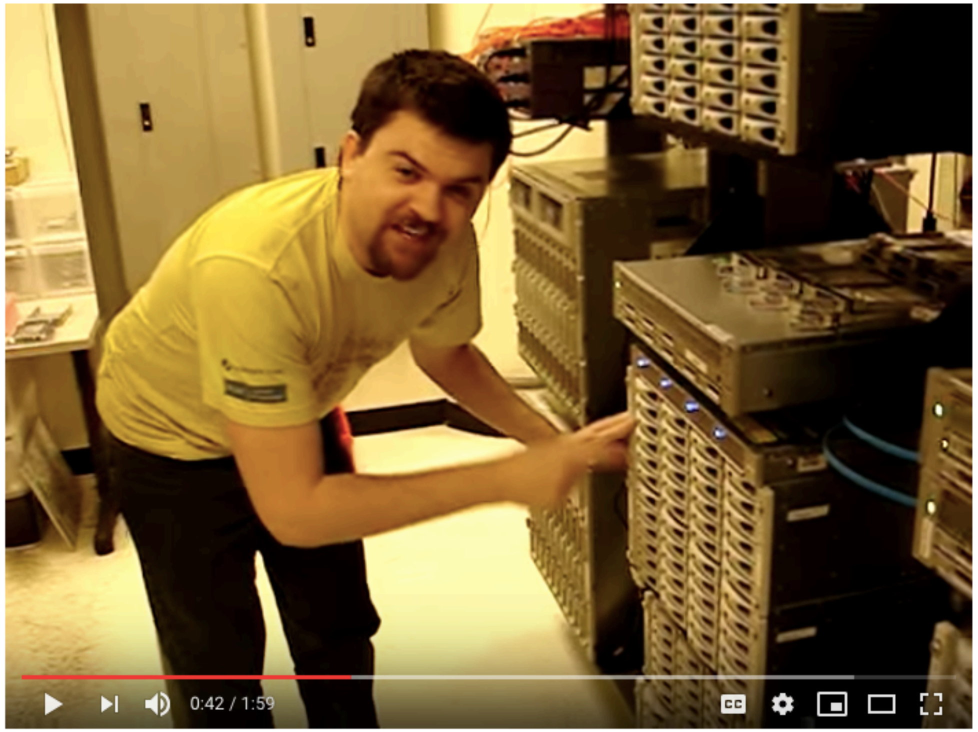

December 2008

# We know vibration hurts hard disks!


December 2008

# We know vibration hurts hard disks!

YouTube




Shouting in the Datacenter

1,671,897 views

9.6K 111 SHARE SAVE

**December 2008**

## Yes, there are fixes.

HPC wire

Since 1987 - Covering the Fastest Computers in the World and the People Who Run Them

Startup Takes Aim at Performance-Killing Vibration in Datacenter

By Michael Feldman

January 19, 2010

## But, they are expensive!

The price premium for a Green Platform rack compared to a traditional metal rack is significant. According to Gordon, an AVP will cost four to five times that of a steel rack (which runs around $2,000). But to Gordon, that's not the way to look at this solution. Since the AVP-1000 improves performance and lowers energy costs, the rack can pay for itself in less than 12 months — sometimes

https://www.hpcwire.com/2010/01/19/startup_takes_aim_at_performance-killing_vibration_in_datacenter/

# Because....


**SSDs are higher performant and do not have moving mechanical parts.**

**Now, SSDs are operating in increasingly vibration-prone environments!**


Data Centers


Self-Driving
Cars
Self-Driving
48
mph

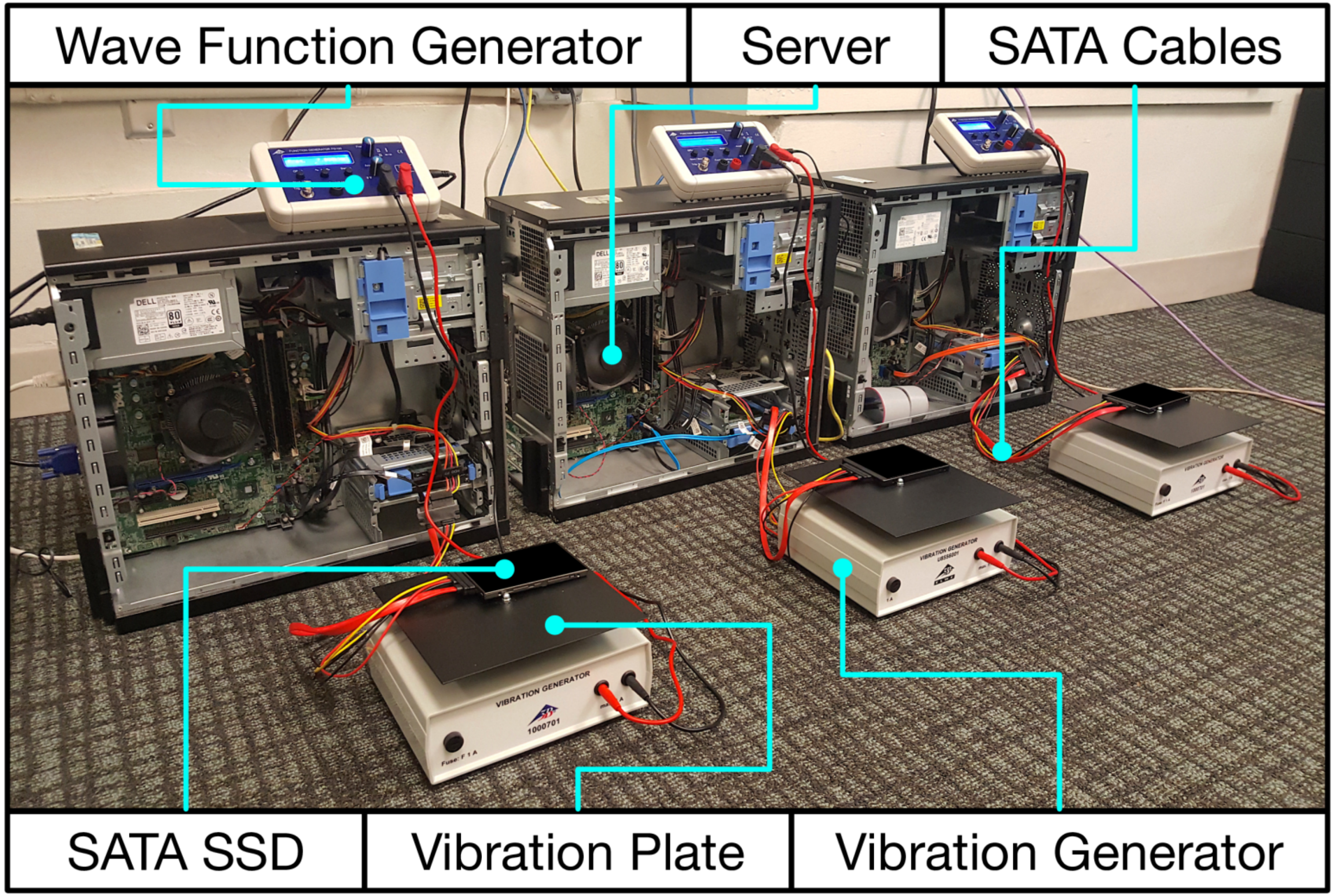

Battlefield


Space
Explorations

# Time to repeat what Brendan Gregg did to hard disks in 2008, but this time to SSDs?

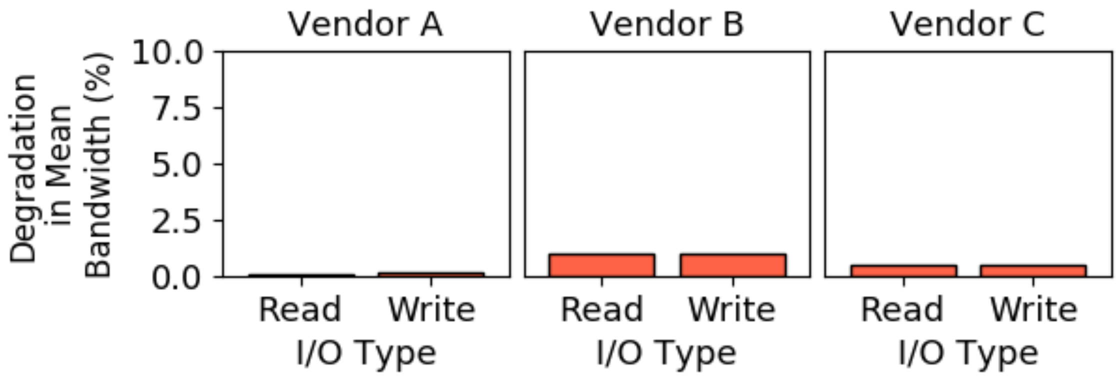
YouTube

# Perhaps, a bit more scientific and controlled!

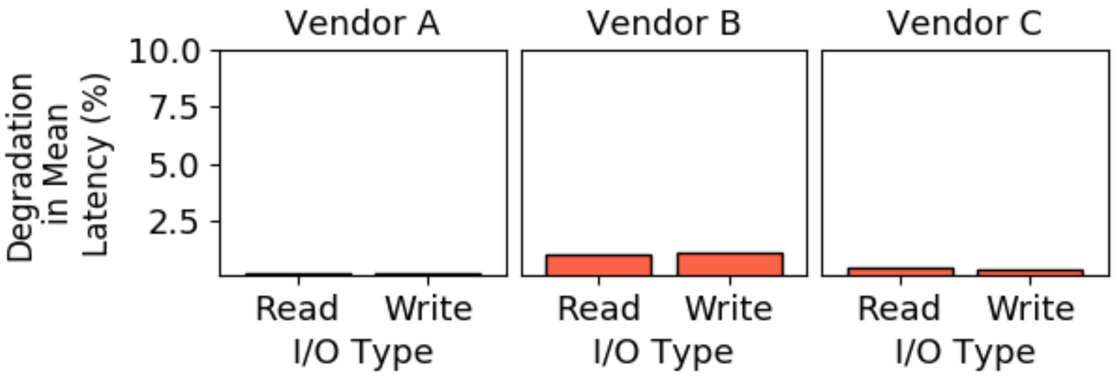


**Vibration intensity lower than the vendor-specified limits!**

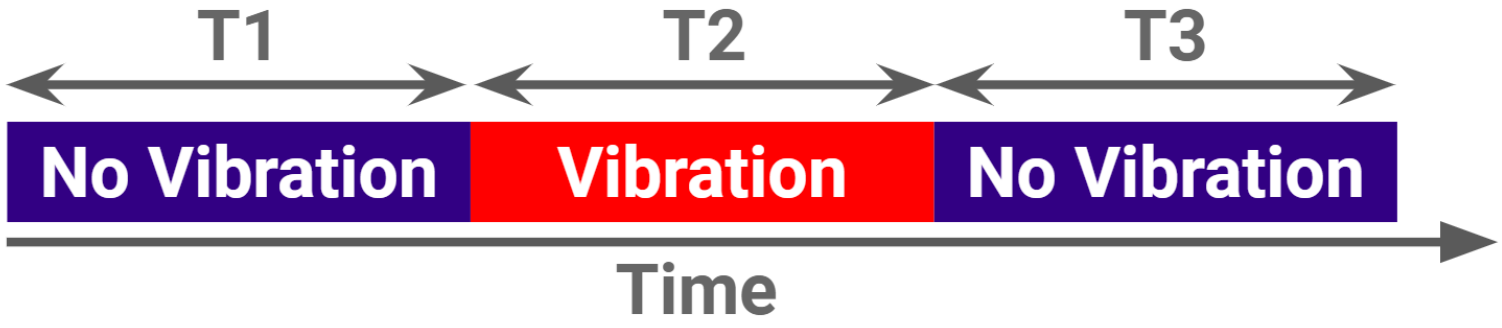
# As the conventional wisdom would suggest, vibration does not seem to have any visible effect on SSD performance!




Effect on mean I/O bandwidth

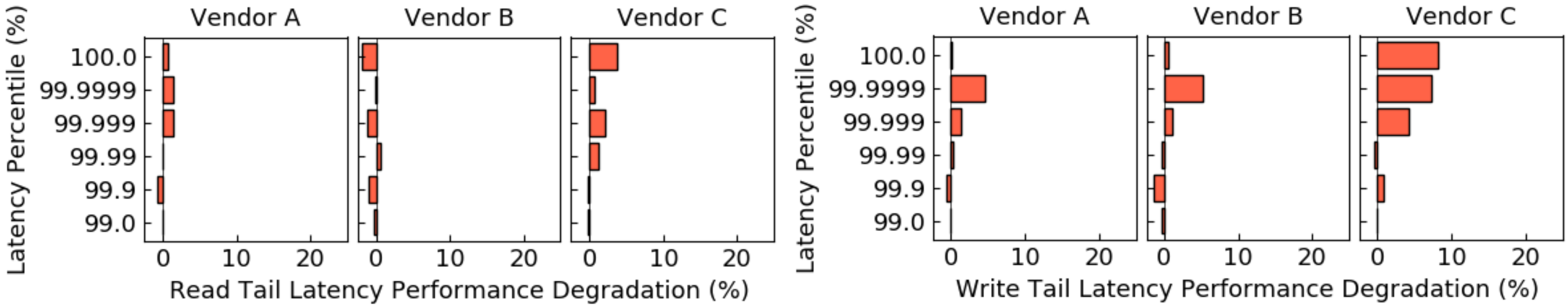



Effect on mean I/O latency



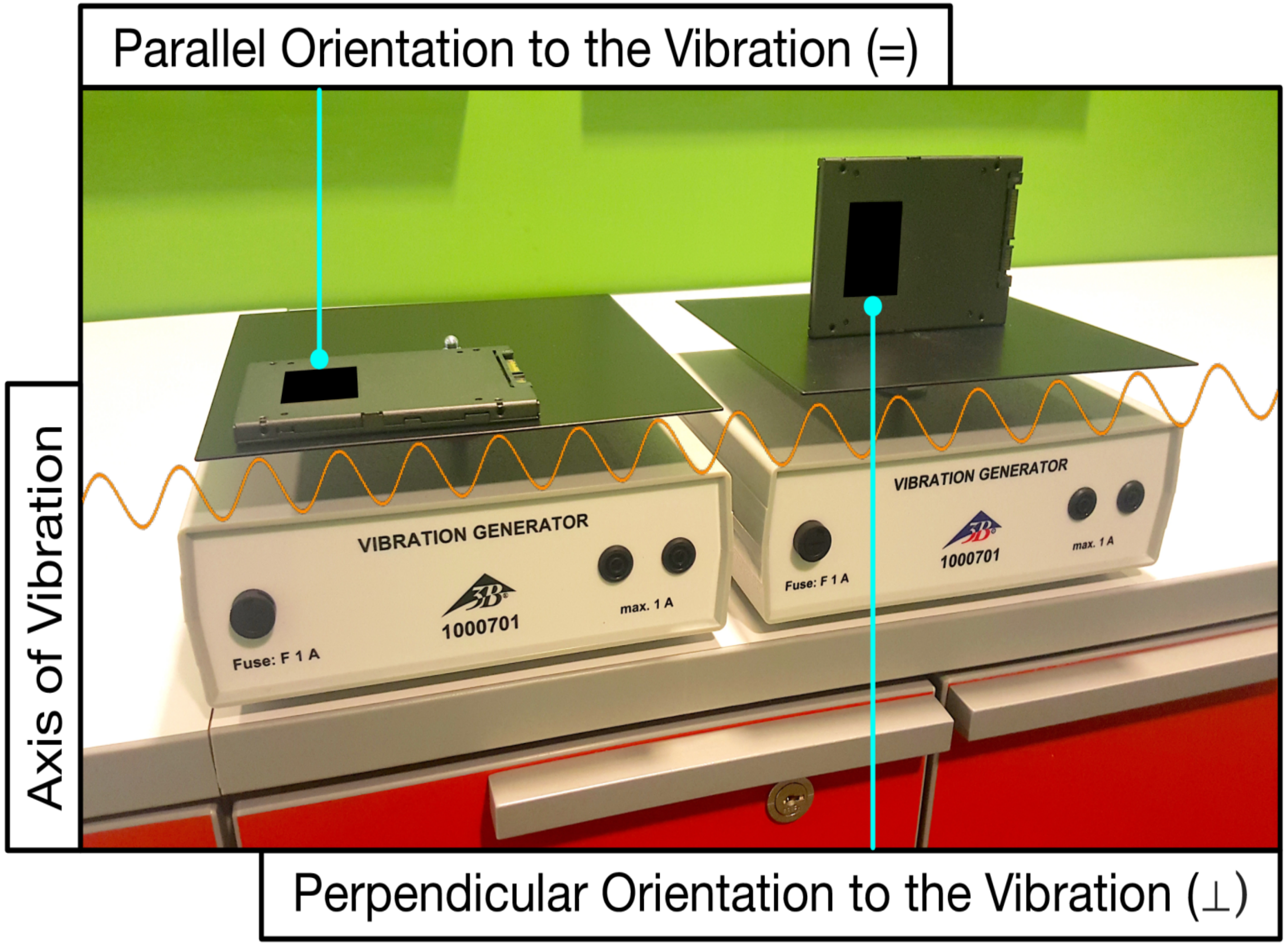
**But, when we dig deeper…**

# Vibration can affect the I/O tail latency significantly!

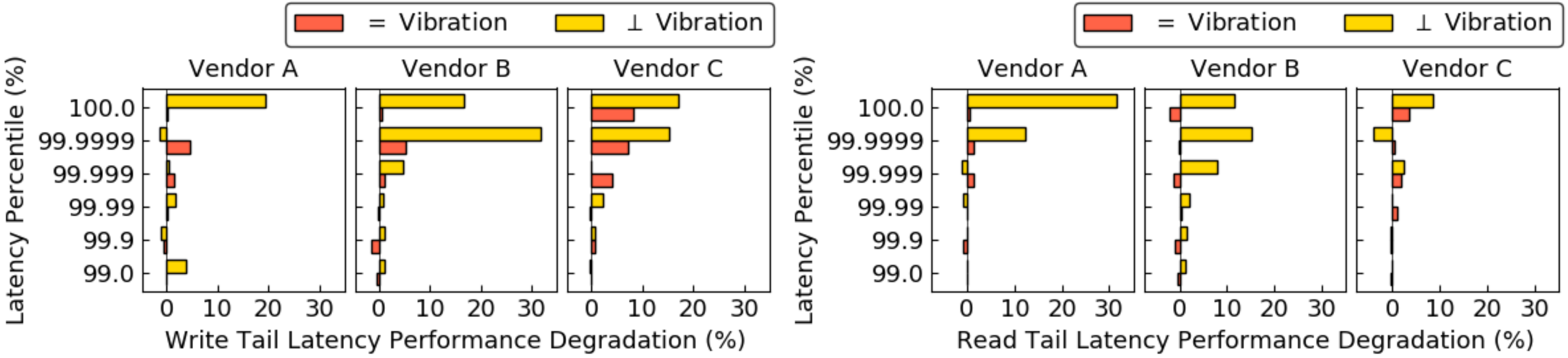



**Tail latency degraded by up to 10% across vendors and I/O type (read and write).**

# Axis of Vibration



# Axis of Vibration Matters a Lot!




**Effect of ⊥ vibration on tail latency is much worse than = vibration, up to 30% in some cases!**

**I/O tail latency gets worse under <u>active vibration</u> across vendors and I/O types, and the magnitude may depend on the <u>axis of vibration</u>!**

**I/O tail latency gets worse under <u>active vibration</u> across vendors and I/O types, and the magnitude may depend on the <u>axis of vibration</u>!**

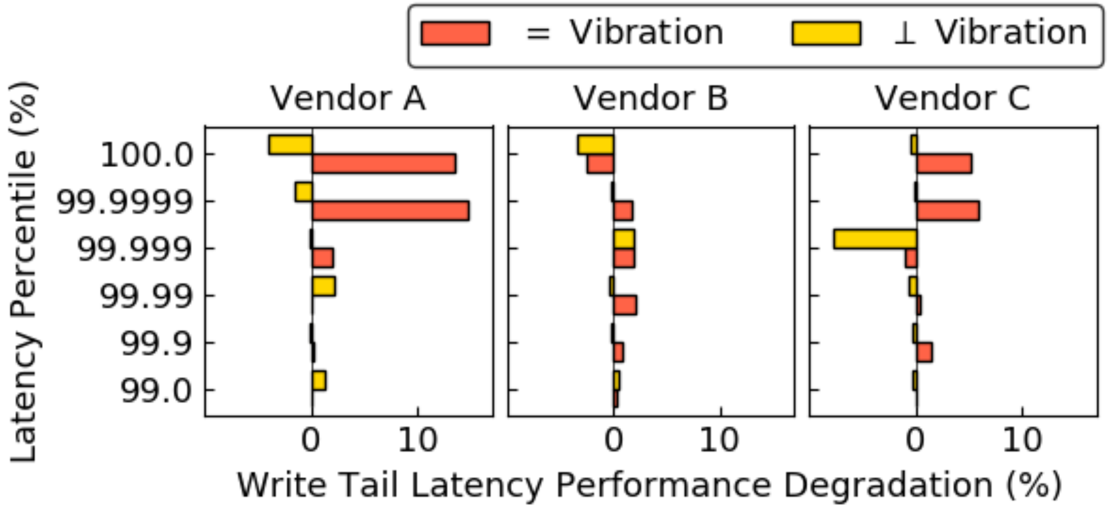
Then, all I need to do is not operate under active vibrations, just like hard disk days!

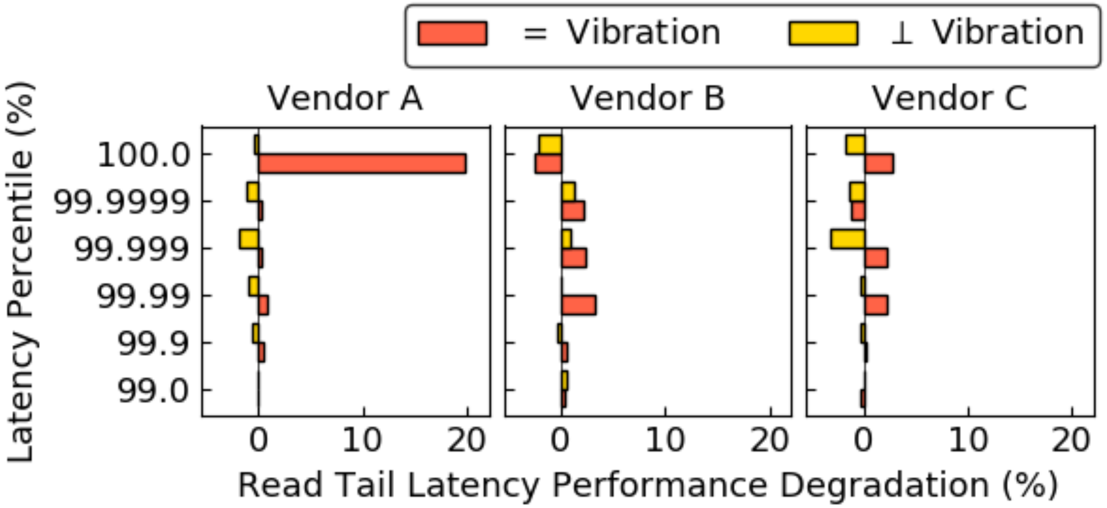
Unfortunately, no!

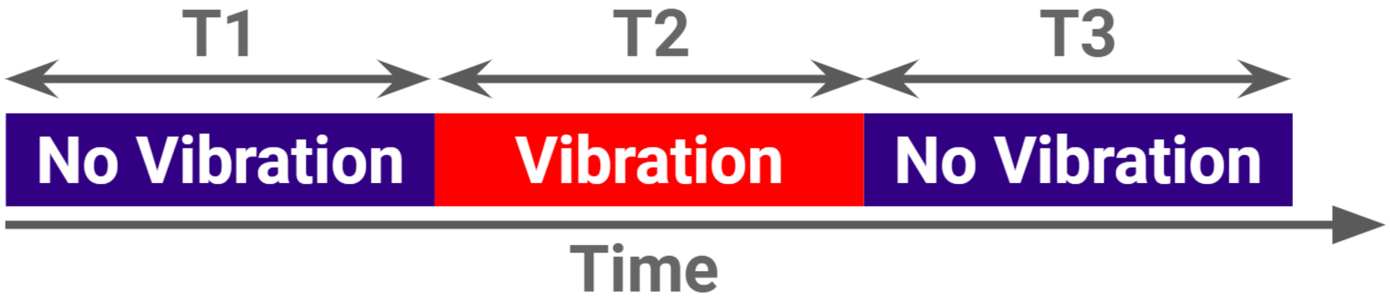
Vibration effects on SSDs tend to persist.

Nature and magnitude of post-effects depends on the length of the vibration!

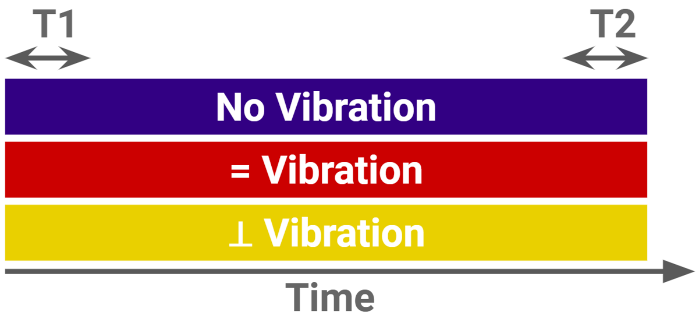
# Short-term Vibrations Can Leave Permanent Post-effects on Tail Latency!

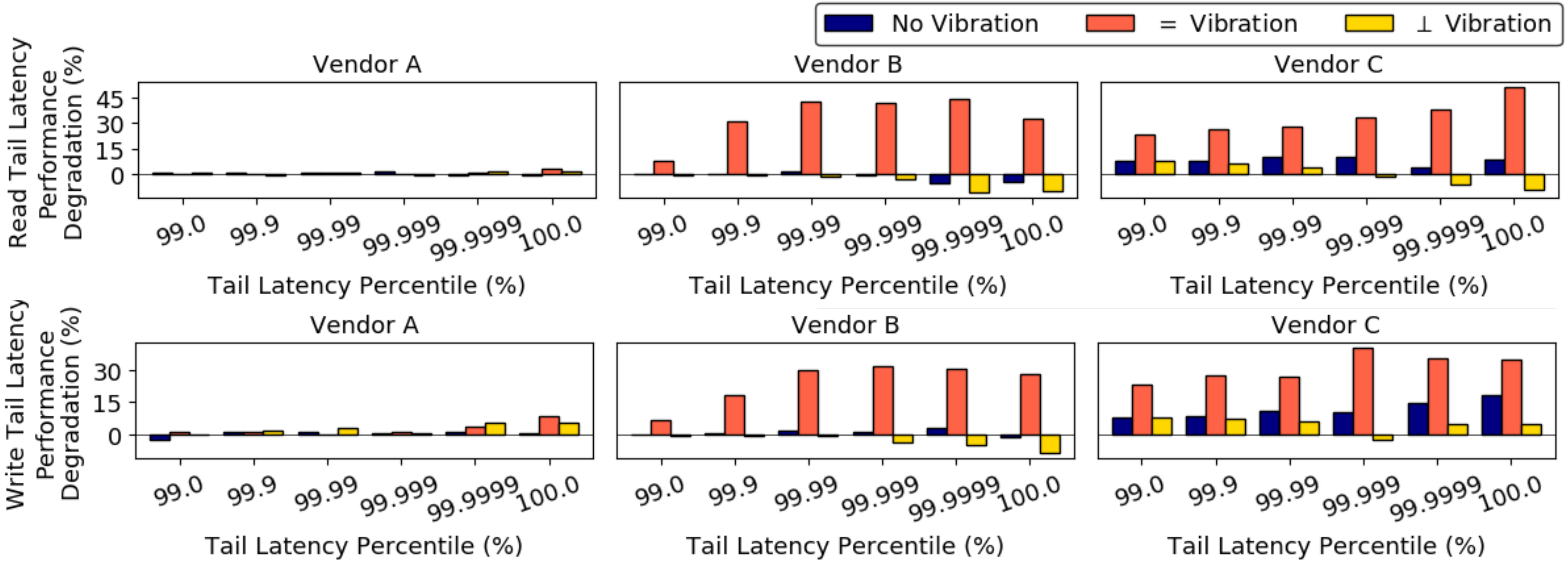


# Long-term vibrations are even more harmful!




**Long-term exposure to vibration can degrade the tail latency by as much as 45%!**

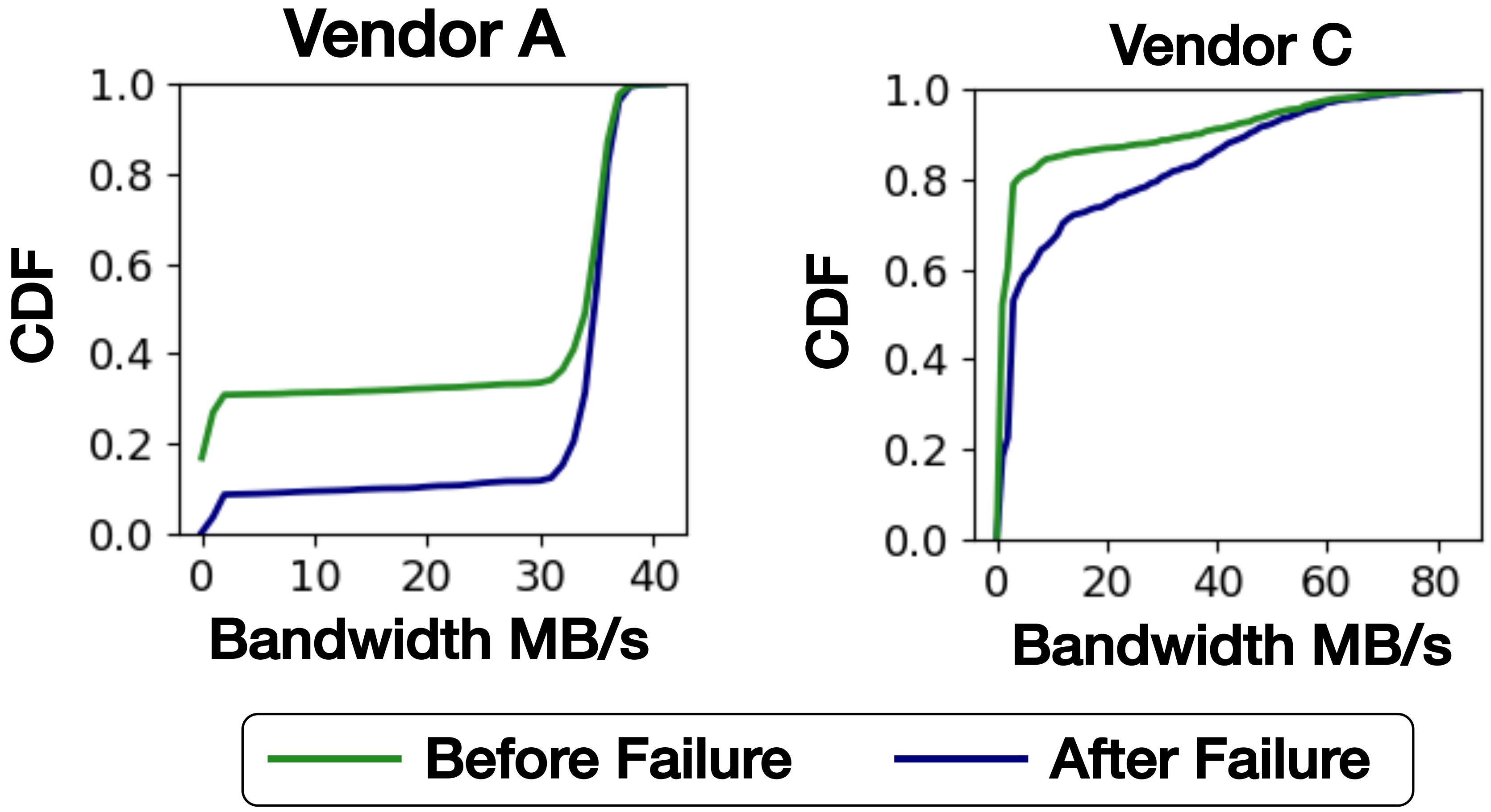
# Surprisingly, long-term vibrations can also lead to SSD failures !

Some SSDs operating under vibration observed silent and transient failures soon after the end of the long-term window, but before reaching their write-endurance limit. These SSDs functioned correctly after a restart, until the next failure.

```
kernel: [1209891.438012] sd 0:0:0:0: [sda] Synchronizing SCSI cache
kernel: [1209891.438033] sd 0:0:0:0: [sda] Synchronize Cache(10) failed: Result:
                                           hostbyte=DID_BAD_TARGET driverbyte=DRIVER_OK
kernel: [1209891.438034] sd 0:0:0:0: [sda] Stopping disk
kernel: [1209891.438038] sd 0:0:0:0: [sda] Start/Stop Unit failed: Result:
                                           hostbyte=DID_BAD_TARGET driverbyte=DRIVER_OK
system-udevd[28027]: Process '/lib/udev/hdparm' failed with exit code 5.
```

**Failures prone to be classified as NDFs (No Defect Found)**

# These failures result in permanent performance degradation!



SSD vibration effects tend to persist even if the length of exposure to vibration is short!

Long-term vibrations can degrade both the tail I/O latency and bandwidth.

Long-term vibrations can also lead to silent failures and permanent bandwidth degradation.

# Conclusion

**Vibrations considered harmful *even* for SSDs!**

**Next time, you borrow or buy an SSD, inquire if the device *was exposed to vibration*, *in what axis*, and *for how long*?**

# Back Up Slides

## *You can't avoid them, you can't predict them, but you can choose who gets hit by them!*

With sincere thanks to students and collaborators at Oak Ridge National Lab, Lawrence Berkeley National Lab, Argonne National Lab, Northeastern University, College of William & Mary and Wayne State University

Northeastern University