Aarohi: Making Real-Time Node Failure Prediction Feasible

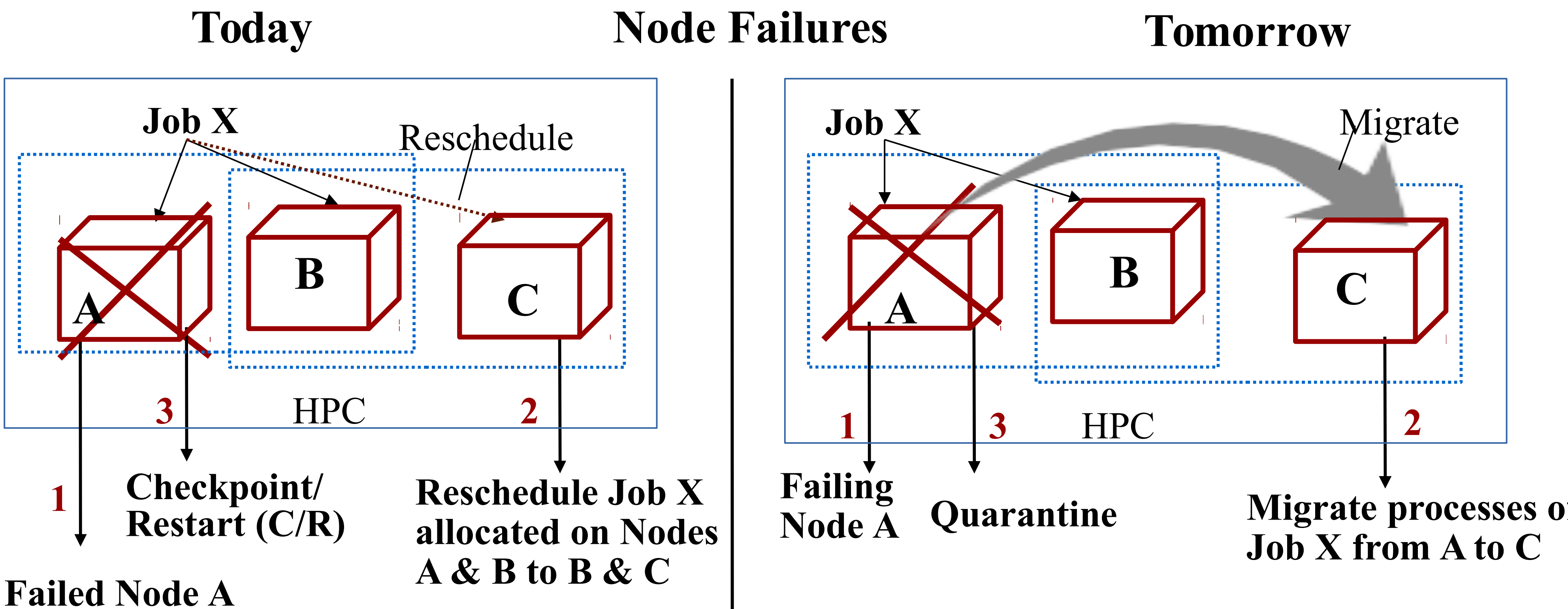
Anwesha Das[1], Frank Mueller[1], Barry Rountree[2]

[1]North Carolina State University
[2]Lawrence Livermore National Lab

Computer Science NC STATE UNIVERSITY

Lawrence Livermore National Laboratory

Proactive Fault Tolerance



- Unprecedented HPC failures cause computation and power wastage
- Reactive approaches (e.g., C/R) can be expensive incurring high overhead

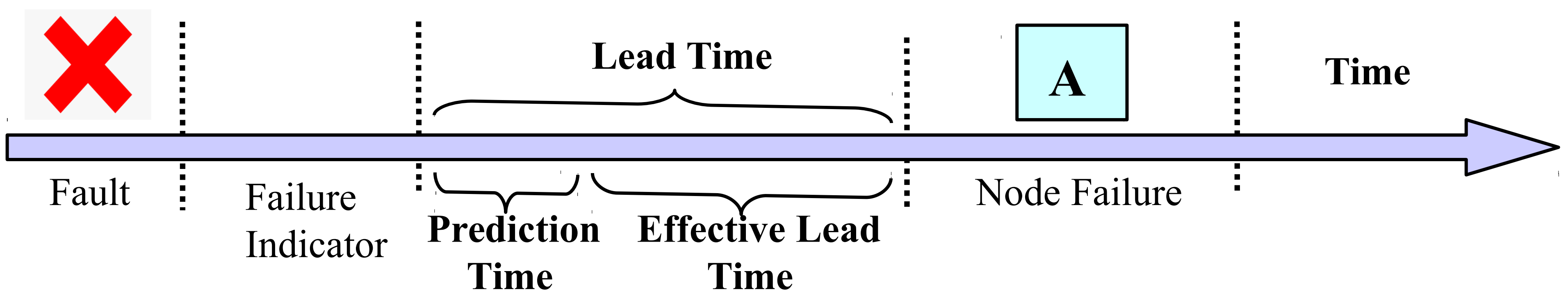
- Can we predict node failures fast enough to enable proactive recovery approaches (e.g., migration)?
- Reduce compute capacity wastage, save production time, and power

HPC Failure Prediction

- Studies on HPC Failure Analysis → Characterize/Detect/Predict Failures
  - Lead time estimates: ~2 to 22 minutes [Klinkenberg et al. Cluster'17, Das et al. HPDC'18]
  - Feasible proactive recovery approaches: Quarantining, Not scheduling jobs on unhealthy nodes, process cloning, job migration (< 24 seconds) etc.
  - Existing schemes: Offline training on voluminous data with high accuracy
    - Less studies on online prediction time analysis (on test data) w.r.t. lead time

- Real–Time Failure Prediction → Offline Training to Online Testing
  - After offline training, achieve speedup during testing
  - Rapid inference considering low MTBFs (mean time between failures) and dense log messages
  - Provide support for ML-oriented failure analysis methods
  - Facilitate proactive resilience by saving computation, production time, and energy

Aarohi strives to achieve low prediction times during testing to retain sufficient effective lead times to failures !!



Background and Challenges

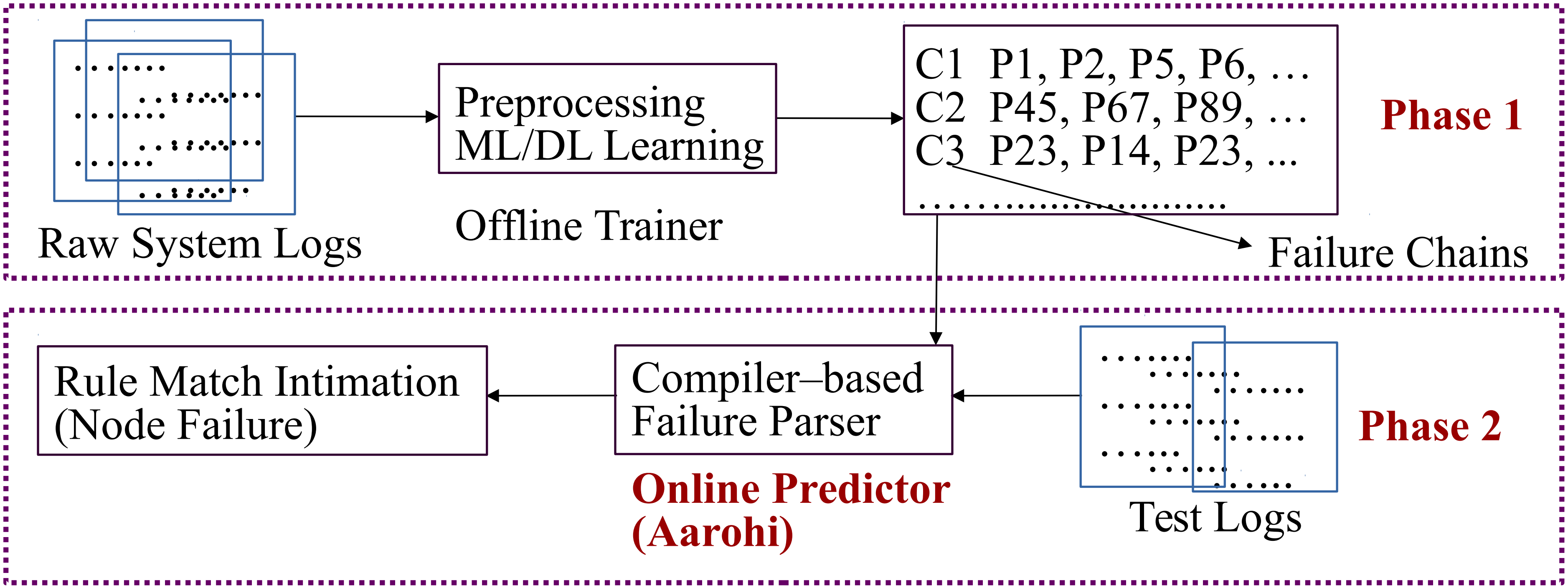
- HPC failure prediction efforts by mining system logs
  - Machine learning (ML)-based predictors proposed with lead time estimates (e.g., 2 to 3 minutes) for production machines (e.g., Cray supercomputers)
  - Most solutions have effective offline trainers (e.g., > 80% recall or accuracy)
  - Less studies consider inference speed for practical usage

- Challenges for real-time failure prediction
  - **ML-based training**: Not necessarily fast enough for real-time deployment
  - **Prediction times**: Compatibility with log message inter-arrival times, While testing messages need not be in batches (unlike training, analyze each incoming phrase)
  - **Proactive actions:** Recovery may not be completed unless sufficient effective lead time (considering prediction time) remains during analysis

- Reusability of inference schemes
  - Support for cross system portability
  - Adaptive to system evolution and logging paradigm updates
  - Minimal overhead with system upgrades, not receding its efficacy over time

Node Failure Prediction

- Builds on prior work (e.g., Desh€) having two phases (training and inference)
  - Phase 1: Deep learning (DL)-based training to form failure chains (FCs)
  - Phase 2: Context Free Grammar (CFG) based rapid inference during testing
- **Novelty** is in Phase 2 (this work does not consider training intricacies, i.e., Phase 1)



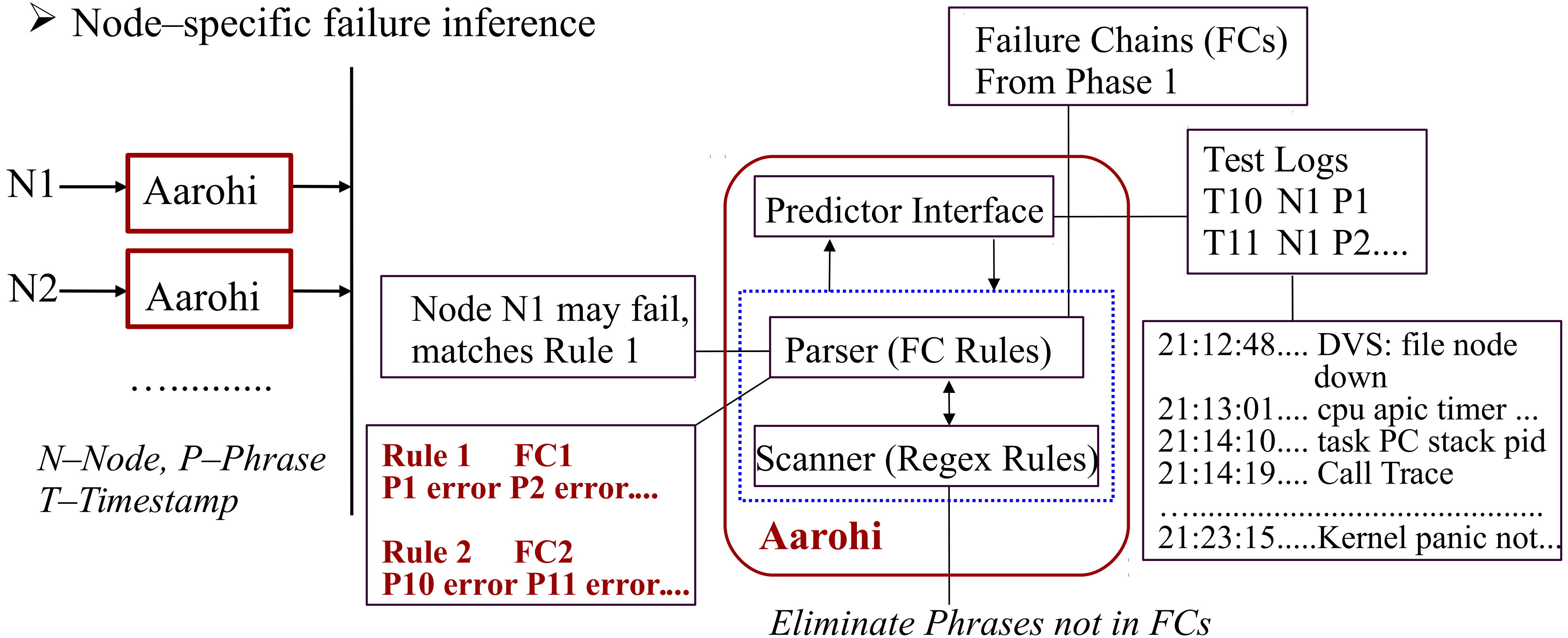
Aarohi Design Goals: Only about inference (Phase 2)

- Generic machine translation of FCs to suitable grammar rules
- Automate predictor generation with available set of FCs (Phase 1 is an apriori for a specific system), minimize rule updates
- Efficient parsing (low execution times) with incoming test logs

€ - Das et al. (HPDC'18)

Aarohi Design

- Real–time inference, process 1 log message at a time (phrase)
- Regular Expression (RE) and CFG based compilation for node failure prediction
- Node–specific failure inference



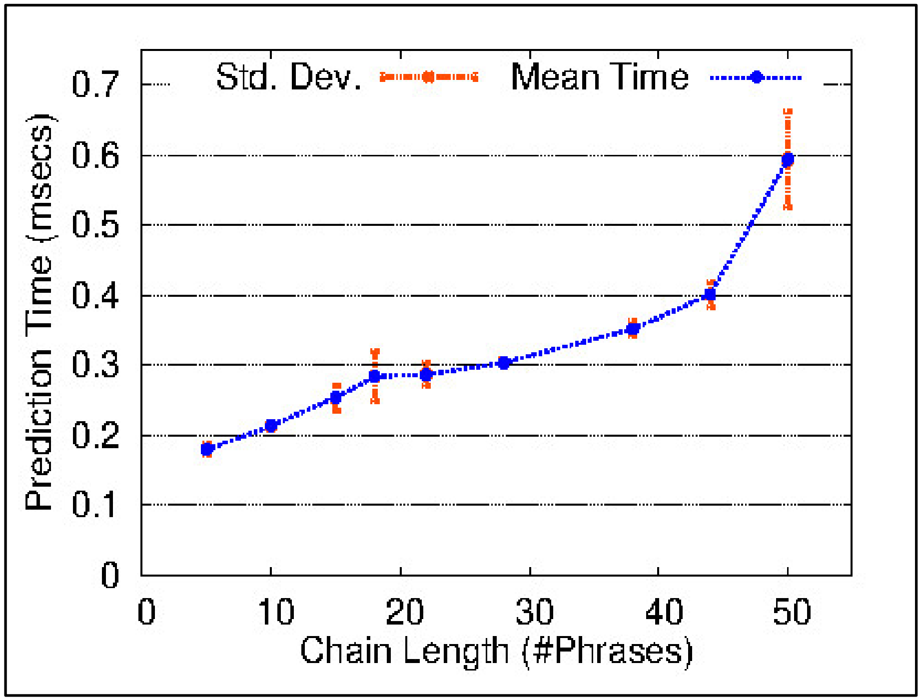
Core Idea:

- Skip irrelevant phrases, tokenize and perform FC–based rule match
- Formulate distinct rules[€] based on specific FCs
- Notify on a rule match and reset the parser based on complete rule matches or threshold based timing violations

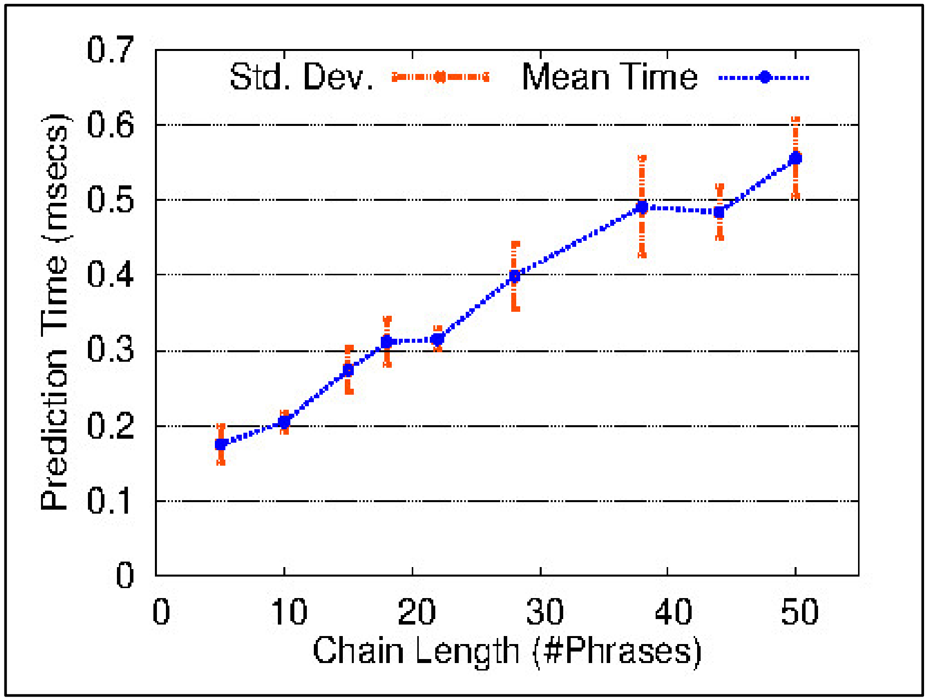
€ - Details in the paper

Chain Prediction Times

- How early in phrase chain (sequence of events) can failure be predicted?



Test logs without benign phrases



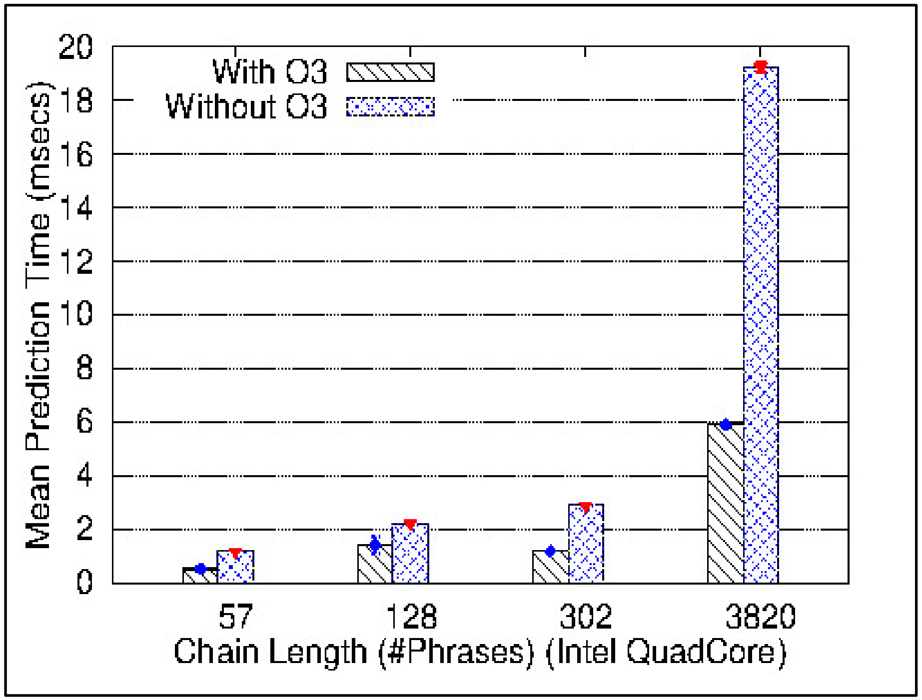
Test logs with benign phrases

Implications:

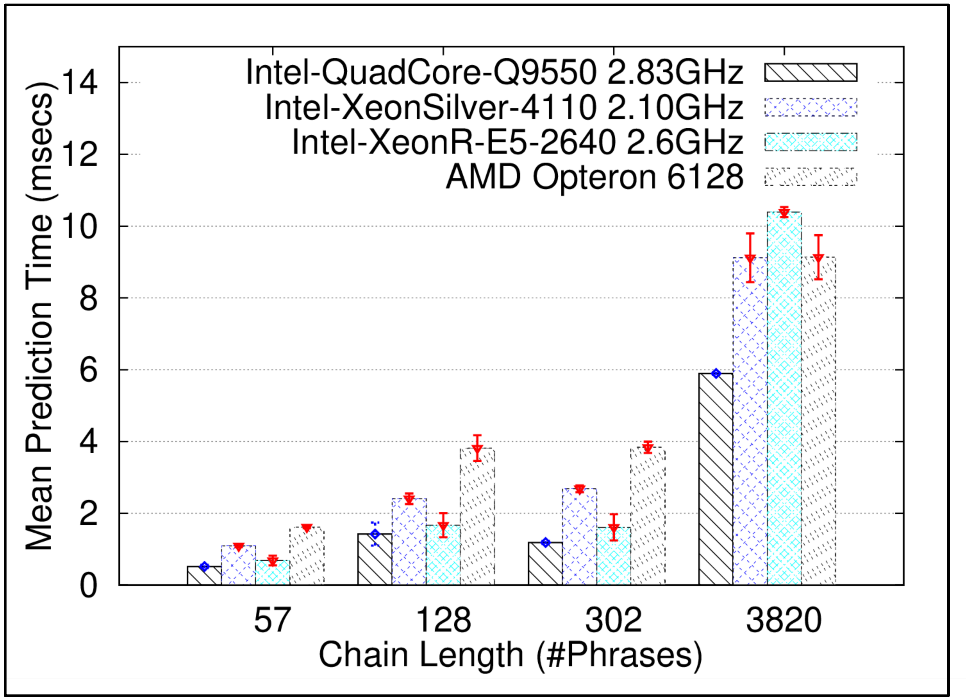
- Increase in chain length (5 to 50 phrases) → Prediction times increase from 0.18 msecs to 0.59 msecs (LHS)
- With benign phrases, prediction times are similar as phrases are skipped (RHS)
- ***Acceptable inference times with phrase inter-arrival times in O(μ or milli-secs) and lead times in O(2 to 3 mins)***

Optimizations and Variations

- How much variations across diverse platforms?



Compiler Flag - O3 Optimization



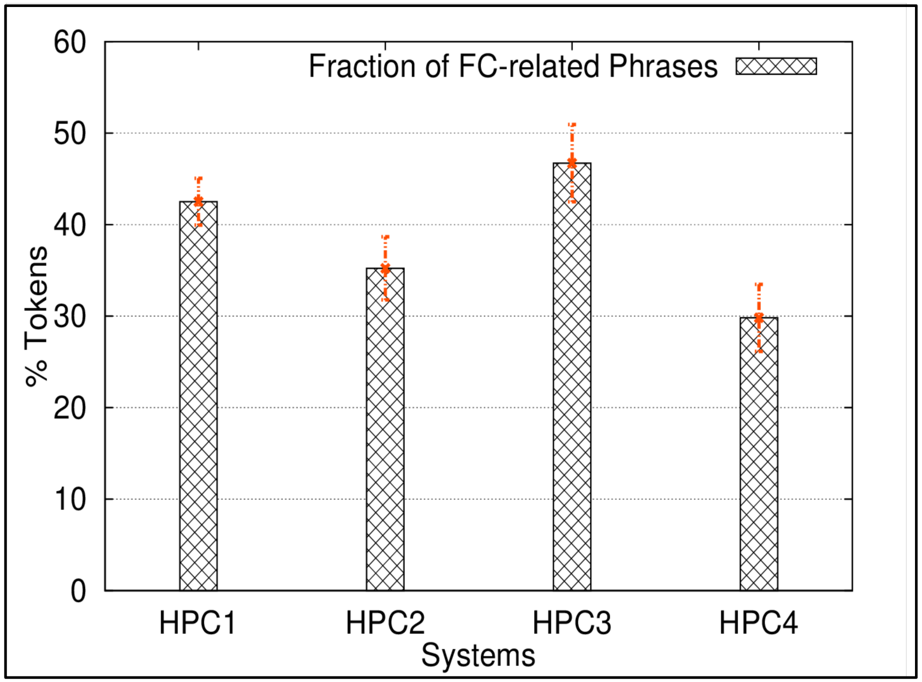
Different CPU architectures

Implications:

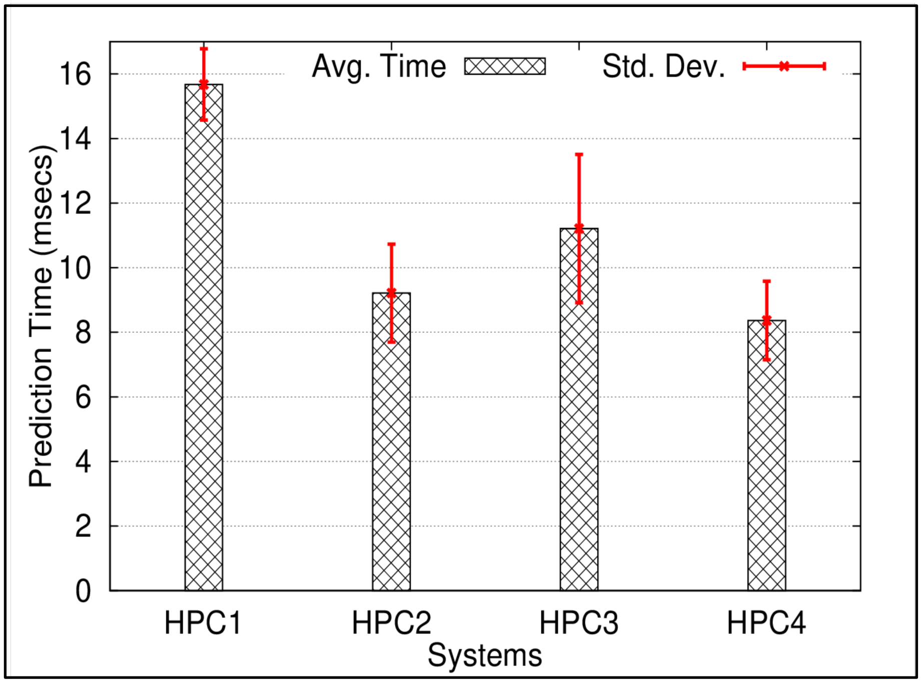
- 128–chain length → 35.8% improvement in inference times with O3 (LHS)
- AMD takes more time in general than Intel, overall prediction times < 11 msecs, standard deviations within ±0.67 msecs (RHS)
- ***Overall inference times are low enough for real–time prediction***

System Prediction Times

- How long are the prediction times across the systems?



Proportion of FC-Related Phrases



System Prediction Times

Implications:

- Prediction times depend on individual phrase sizes (e.g., 4K versus 18K) and proportion of failure related phrases in the test sequence
- During inference → 29.81% to 46.72% of the phrases are FC–related (LHS)
- Overall prediction times across the systems: 8.36 to 15.67 msecs (RHS)
- ***Effective lead times considering prediction times → > 2 to 3 minutes***

Conclusions

- Compiler-based inference scheme:
  - An alternative approach for rapid testing, can provide run-time support to existing ML-based failure prediction methods
- Obtained speedups between 2x to 27.4x over some existing approaches
  - Speedups discernible (higher) with longer sequence of phrases
- Aarohi achieves effective lead times of over 2 minutes, with < 11 milliseconds prediction times (for 3820-length chain)
- Solution is generic, adaptive and consistent over diverse platforms

Predicts imminent node failures in O (sub-seconds) !!

Thank You

Questions?