

# Challenges in predicting failures on the Blue Waters system

Ana Gainaru

Franck Cappello, Marc Snir, Bill Kramer



# Goal

- Online failure prediction
  - On Blue Waters

# So far

- Data mining
  - Different correlation methods
    - Best result: 80% precision and 70% recall
- Signal analysis
  - Used for anomaly detection
    - 87% precision and 51% recall (IPDPS12)
- Hybrid
  - ELSA: Signal analysis with data mining
    - 90% precision and 45% recall (SC12)
    - At least 10 seconds delay

# Failure prediction



- Training
  - Between 3-5 months
- 3 training phases: Parameters influence results

# Failure prediction

training

simulate online



- HELO

Failure prediction is a complex task that requires a deep understanding of the system's behavior. The HELO framework is designed to predict failures by analyzing the system's state and identifying potential issues. This involves a combination of machine learning and rule-based logic to detect anomalies and predict future failures. The HELO framework is used in a variety of applications, including system monitoring and fault diagnosis.

## List of templates

\* errors detected and corrected

node card \* is not fully functional

...

vm: killing process %s n+

# Failure prediction

training

simulate online



## • HELO

HELO (Hardware Error Logging and Online Prediction) is a system designed to detect and predict hardware failures in large-scale computing environments. It uses a combination of hardware sensors and software monitoring to identify potential issues before they cause system downtime. The system is designed to be scalable and can be deployed across a wide range of hardware configurations. HELO provides real-time monitoring and alerting, allowing system administrators to take proactive measures to prevent failures. The system is currently being used in several large-scale computing environments, including the Joint Laboratory for Petascale Computation.

### Parameter

- Cluster goodness
- High values

10 errors detected and corrected  
2 errors detected and corrected

# Failure prediction

training

simulate online



## • HELO

HELO (Heterogeneous Error Loss Optimization) is a machine learning framework for failure prediction. It is designed to handle heterogeneous data and is used to predict failures in a system. The framework is based on a loss function that is optimized to minimize the error rate. It is used to predict failures in a system and is a key component of the HELO framework.

### Parameter

#### • Cluster goodness

#### • Low values

**vm: \* process %s n+**

*vm: killing process %s n+*

*vm: starting process %s n+*

# Failure prediction

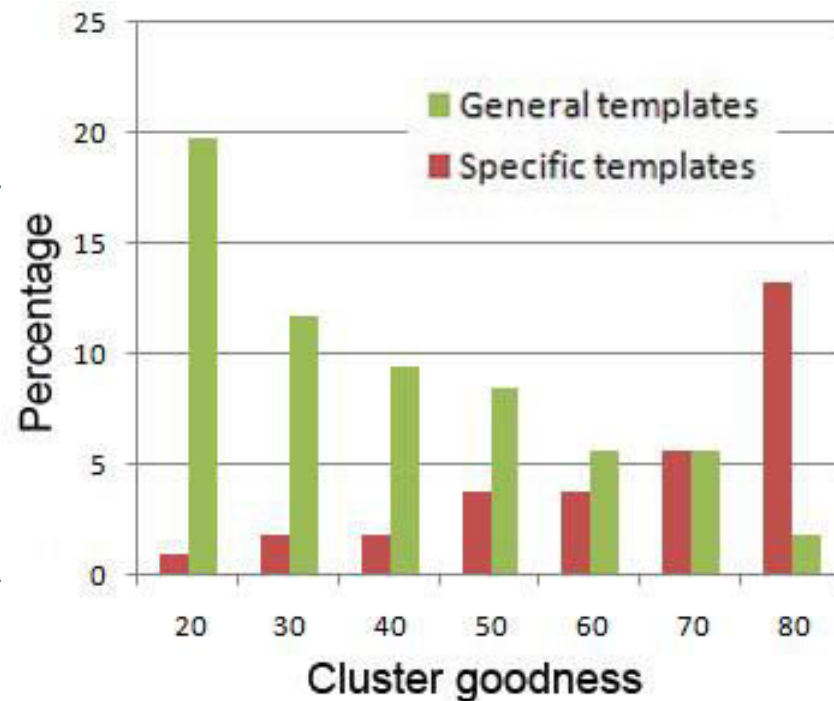
training

simulate online



- HELO

HELO (High-End Learning for Online) is a machine learning framework for failure prediction. It uses a combination of general and specific templates to predict failures. The framework is designed to be scalable and efficient, allowing for real-time simulation and prediction of system failures. It leverages a large dataset of historical failure data to train its models, which are then used to predict future failures in a simulated environment. The framework is implemented in a modular fashion, allowing for easy integration with existing system monitoring and management tools.





# Failure prediction

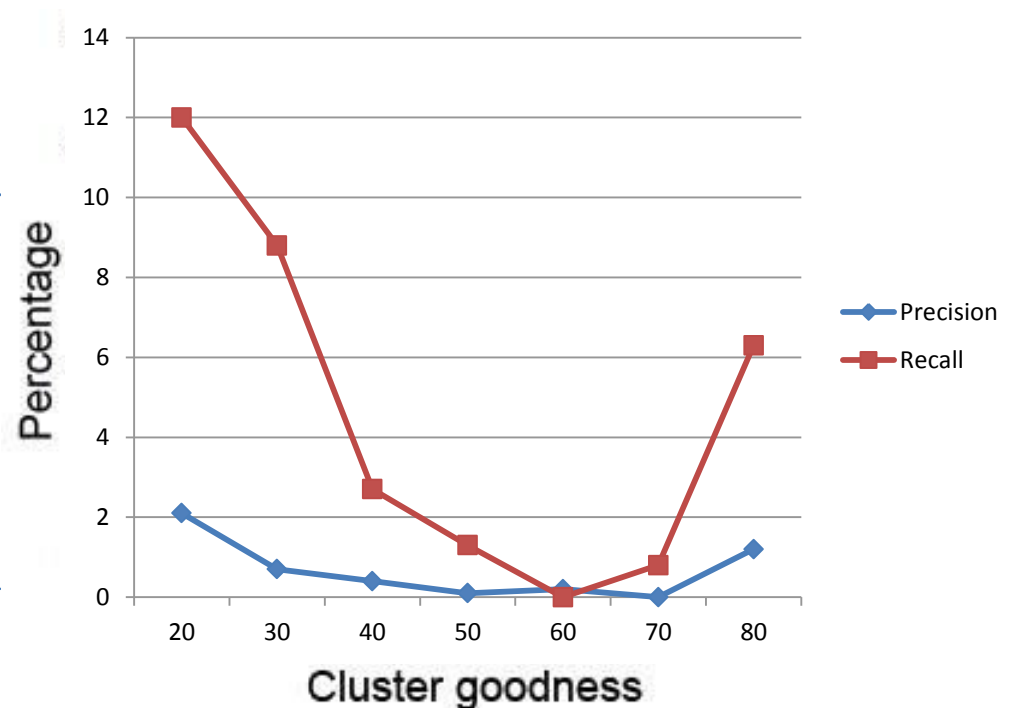
training

simulate online



- HELO

HELO (Heterogeneous Event-driven Load Orchestration) is a framework for managing the execution of applications on heterogeneous hardware. It is designed to be used in a distributed environment, where the application is divided into multiple tasks that can be executed on different hardware components. HELO uses a central controller to manage the tasks and the hardware resources. The controller can be configured to use different scheduling algorithms, such as Round Robin or Shortest Job First. HELO also provides a set of APIs for the application to interact with the controller. The application can use these APIs to request resources, report progress, and receive notifications. HELO is designed to be flexible and extensible, so that it can be adapted to different hardware configurations and application requirements.



# Failure prediction

training

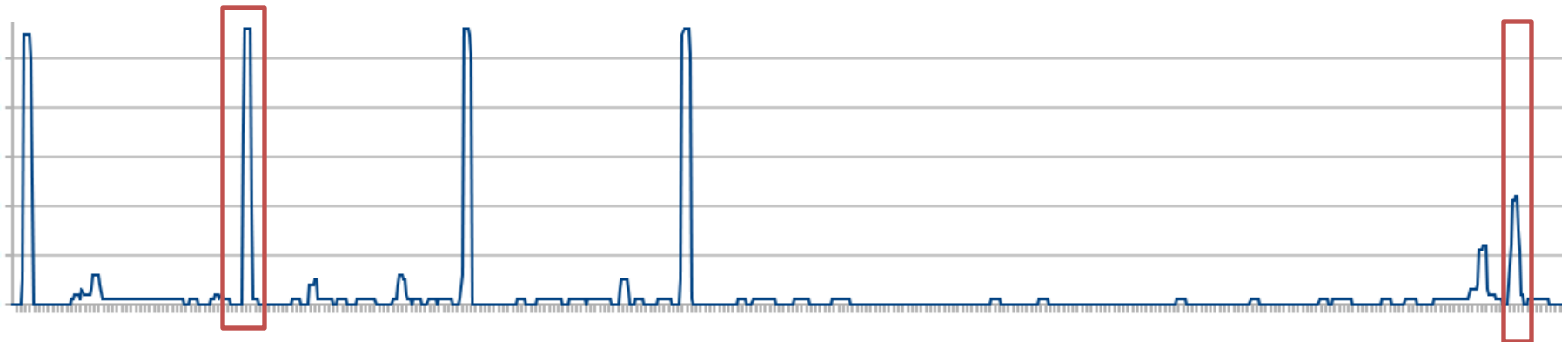
simulate online



- ELSA

Parameter

- Anomaly sensibility



# Failure prediction

training

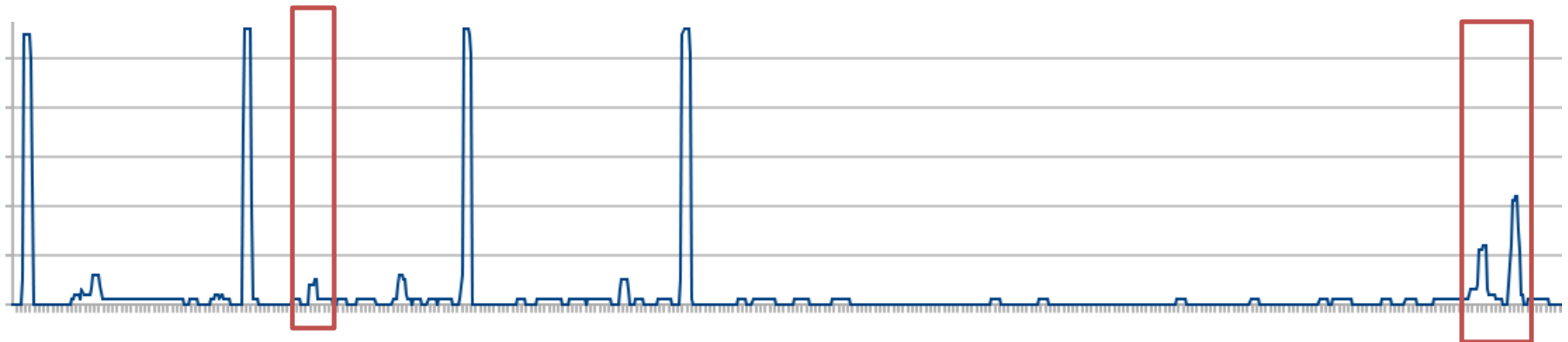
simulate online



- ELSA

Parameter

- Anomaly sensibility



# Failure prediction

training

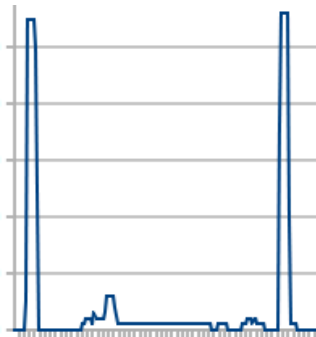
simulate online



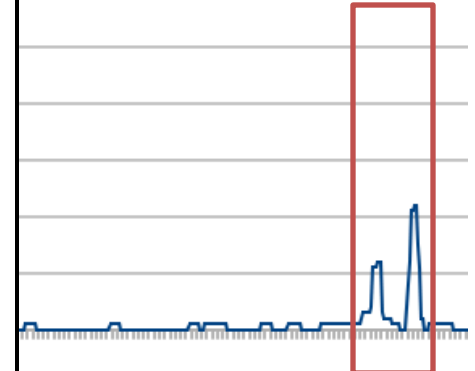
- ELSA

Parameter

- Anomaly sensibility



- Recall varies with 15-18%
- Precision varies with ~10%
- For too many false neg/pos
  - No correlation parameter gives good results



# Failure prediction

training

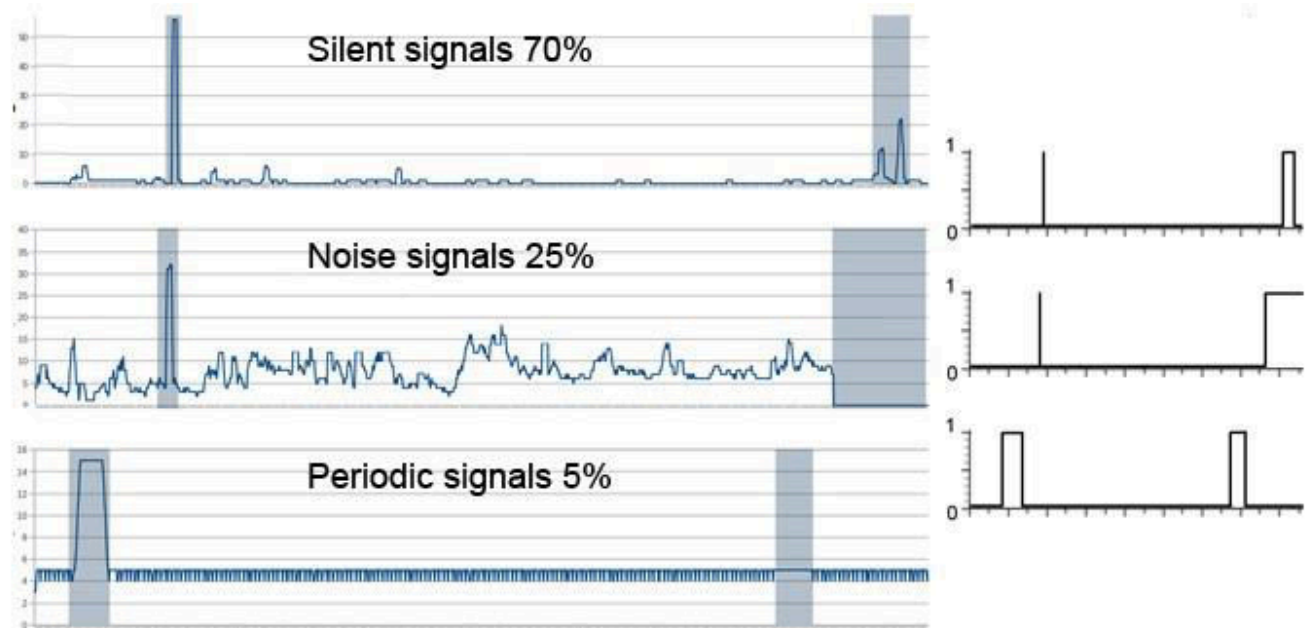
simulate online



- ELSA

Parameter

- Correlation parameters



# Failure prediction

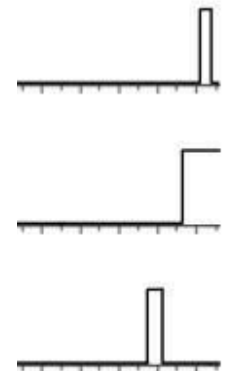
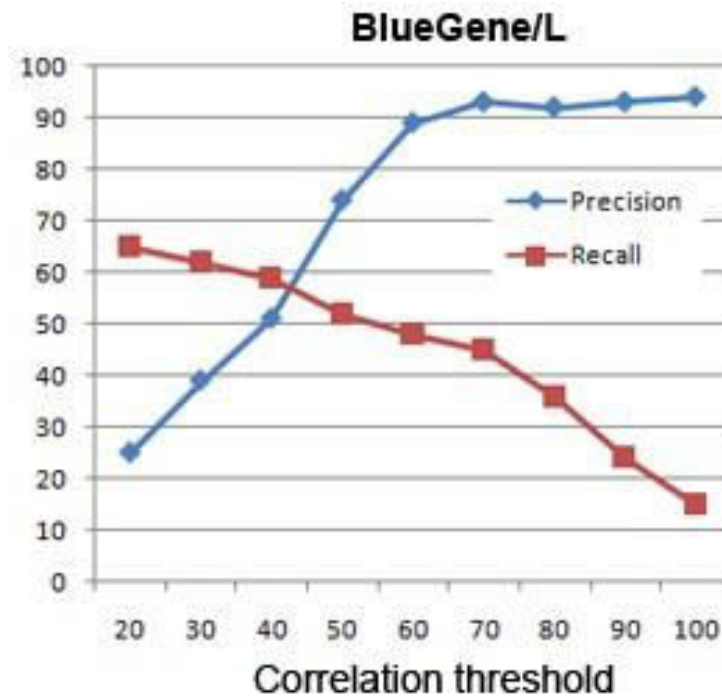
training

simulate online

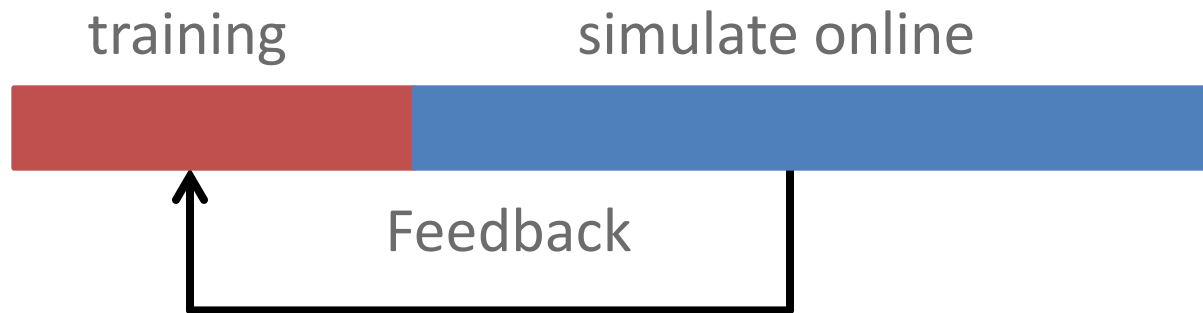
- ELSA

Parameter

- Correlation parameters

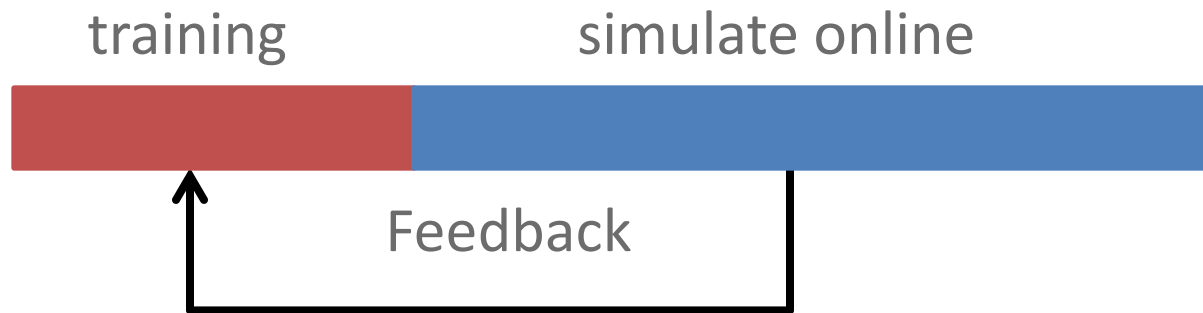


# Failure prediction

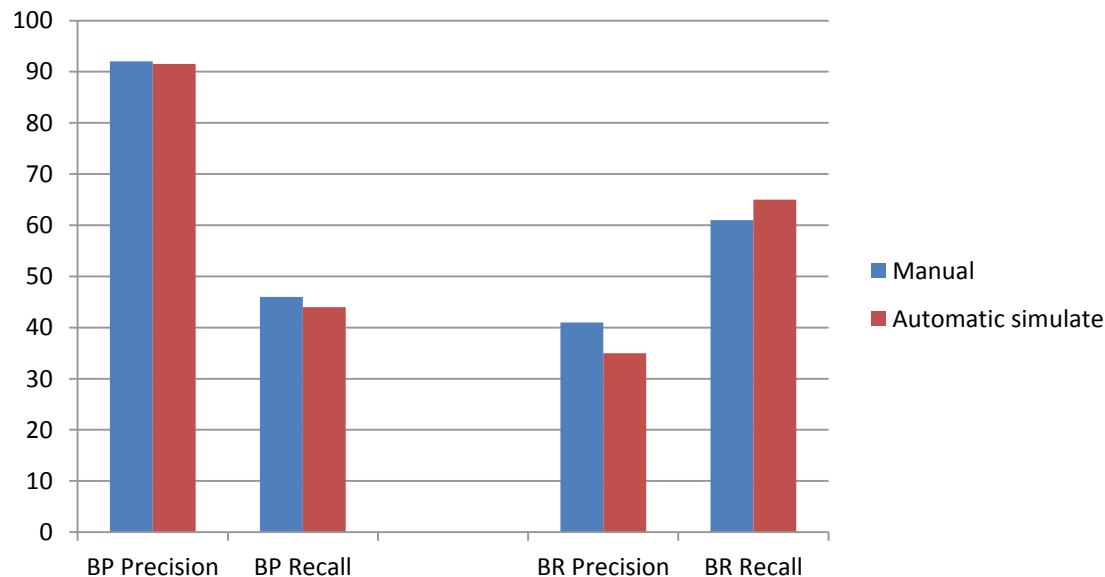


- Simulate online
  - Redo the analysis based on the results
- Automatic tuning of the parameters
  - To get max precision/recall/given relation
  - Brute force

# Failure prediction



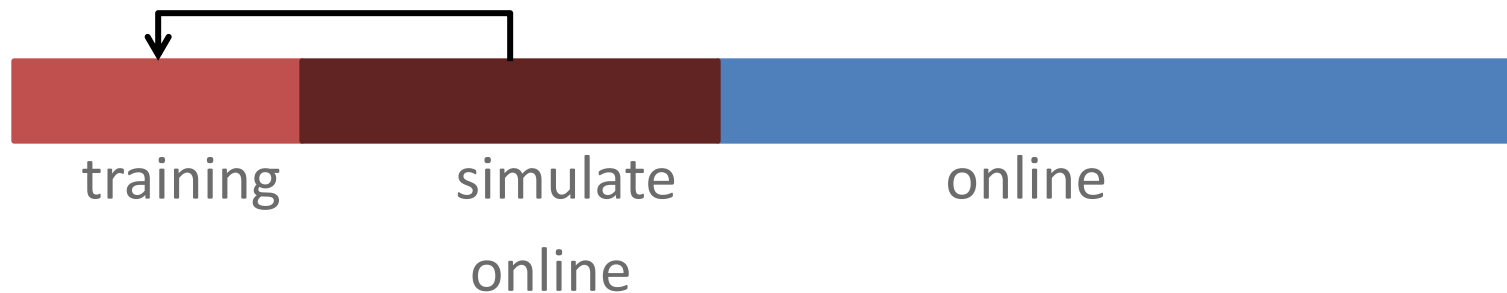
- Simul
- Red
- Autor
- To g
- Bru





# Online prediction

- Choosing parameters
  - Based on experience with other systems
  - or



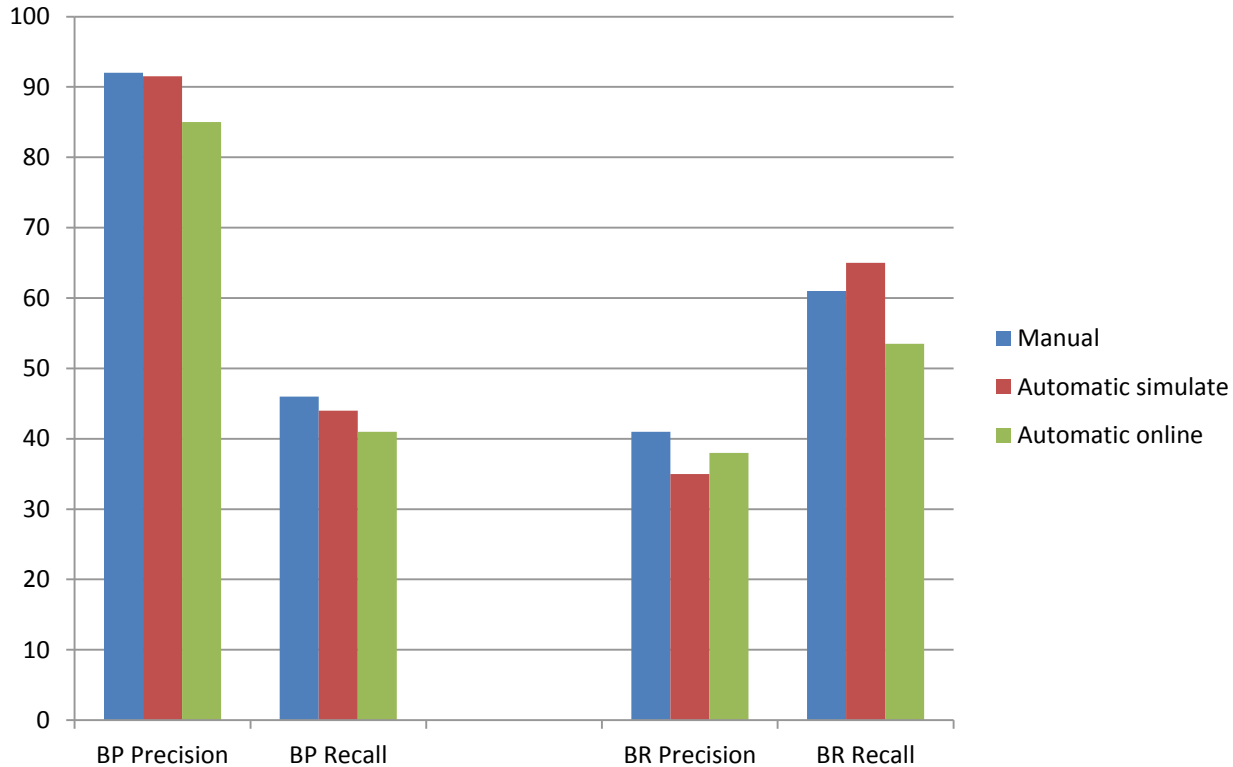
# Online prediction

- Choice

- Baseline

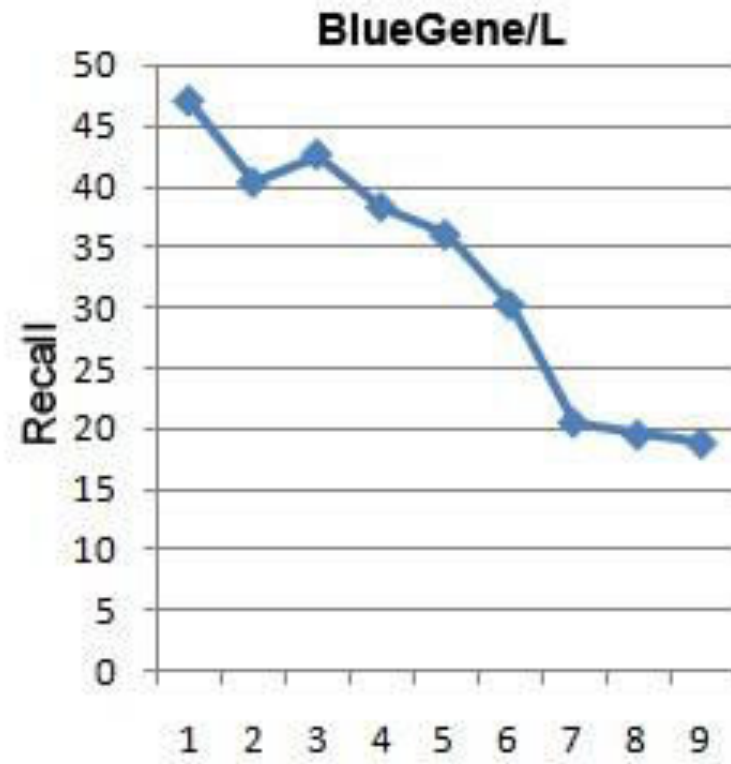
- or

train



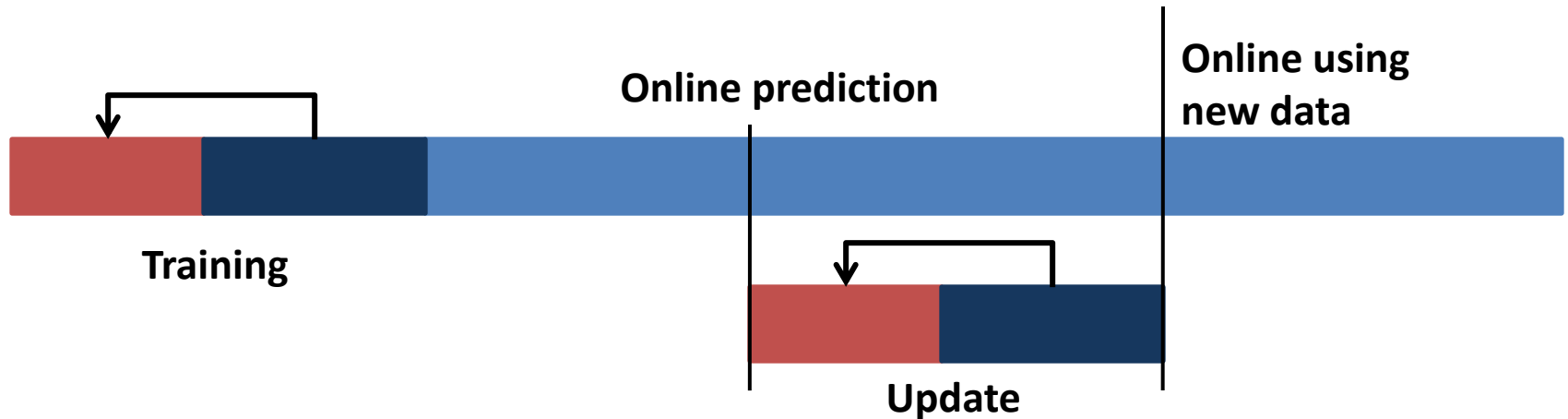
# Online prediction

- Recall on different months



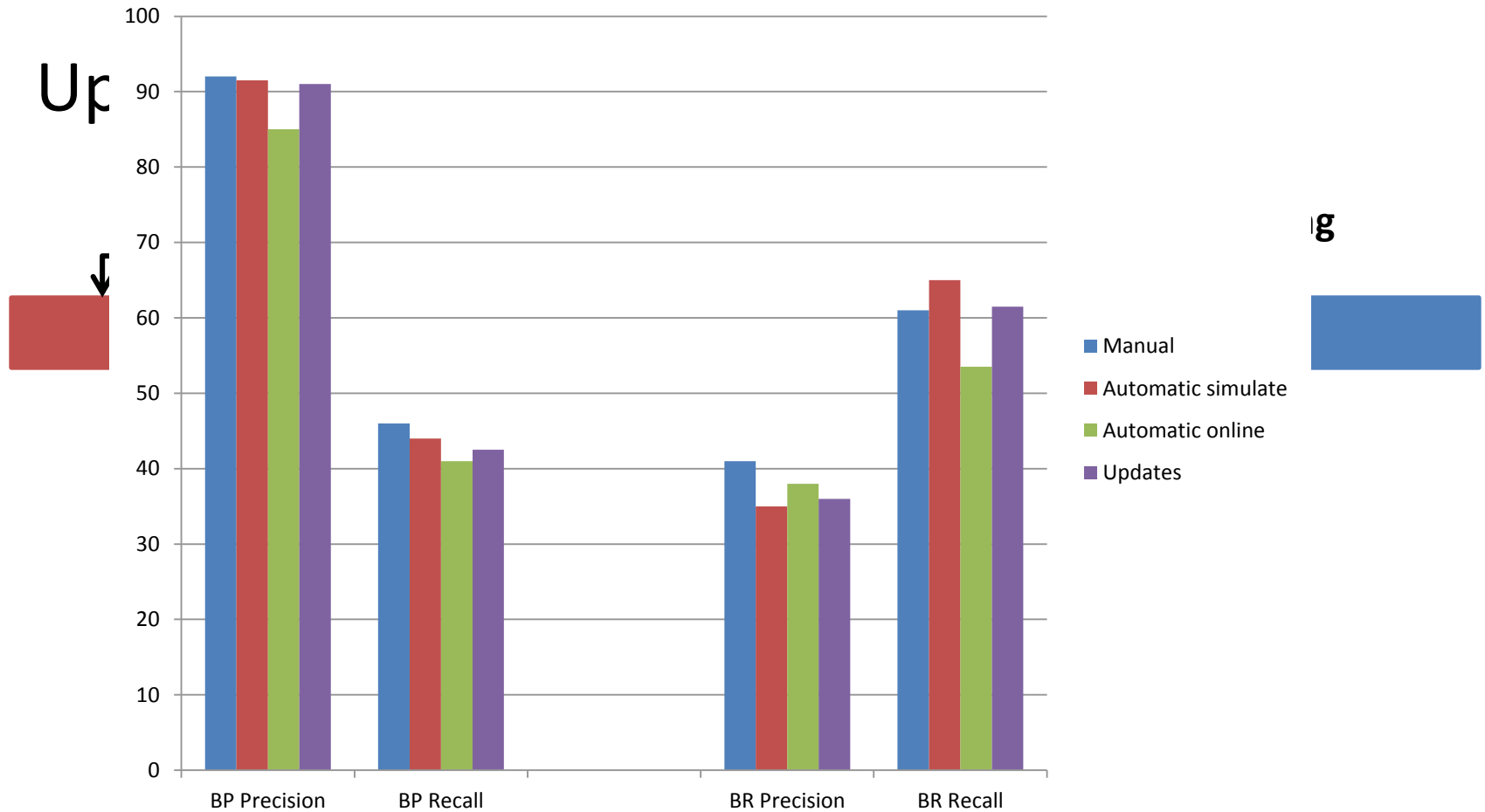
# Online prediction

- Updates



# Online prediction

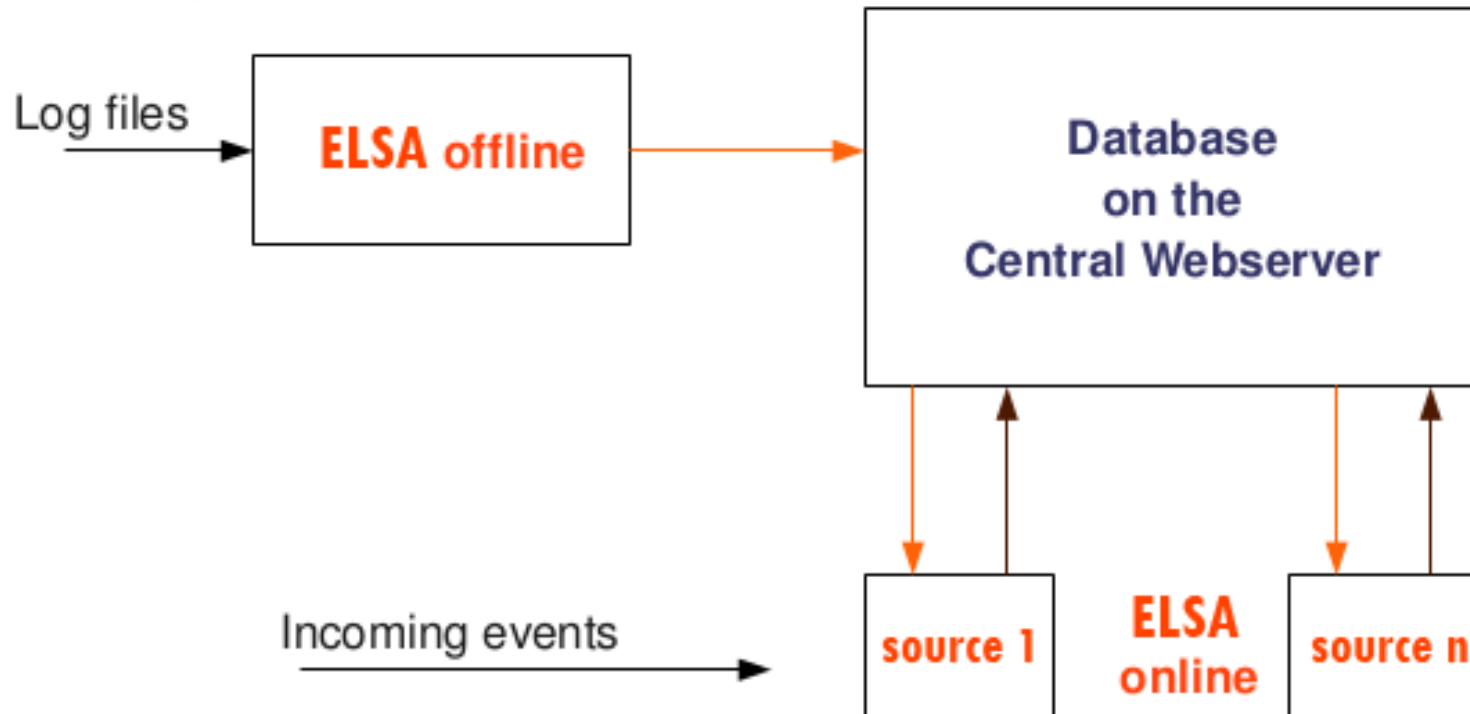
- Up



# Blue Waters

- Event logs are gathered on a separate machine
  - System logs
  - HPSS (High Performance Storage System)
  - Sonexion storage system
  - Moab job scheduler
  - ESMS (Data system manager)
- Failure logs
  - Annotated by Cray
  - Templates identified by NCSA

# Blue Waters



# Blue Waters

- Some numbers

**Table 1: Frequency of Special Characters**

Source	Events/Day	Total Event Types
Syslog	8GB (50mil events)	3,852
HPSS	1MB (900,000 events)	358
Sonexion	3.5GB (10mil events)	3,112
Moab	500 MB (15mil events)	725
ESMS	3GB (12mil events)	2,452
System	Events/Day	Total Event Types
BlueGene/L	5.76MB (25,000 events)	186
BlueGene/P	8.12MB (120,000 events)	252
Mercury	152.4MB (1.5mil events)	563



# Blue Waters

- Some numbers

n – no events t – no templates c – no correlations	HELO	Anomaly detection	Correlation extraction / identification
Complexity	Offline: $O(n \cdot \log n)$ Online: $O(t)$	Offline: $\text{alg} \cdot O(t)$ Online: $\text{alg}$	Offline: $O(t \cdot n \cdot \log n)$ Online: $O(c \cdot \text{avg\_len}(c))$
BGL	~10min	~30min	~8h
Blue Waters	~2h	~5h	>24h

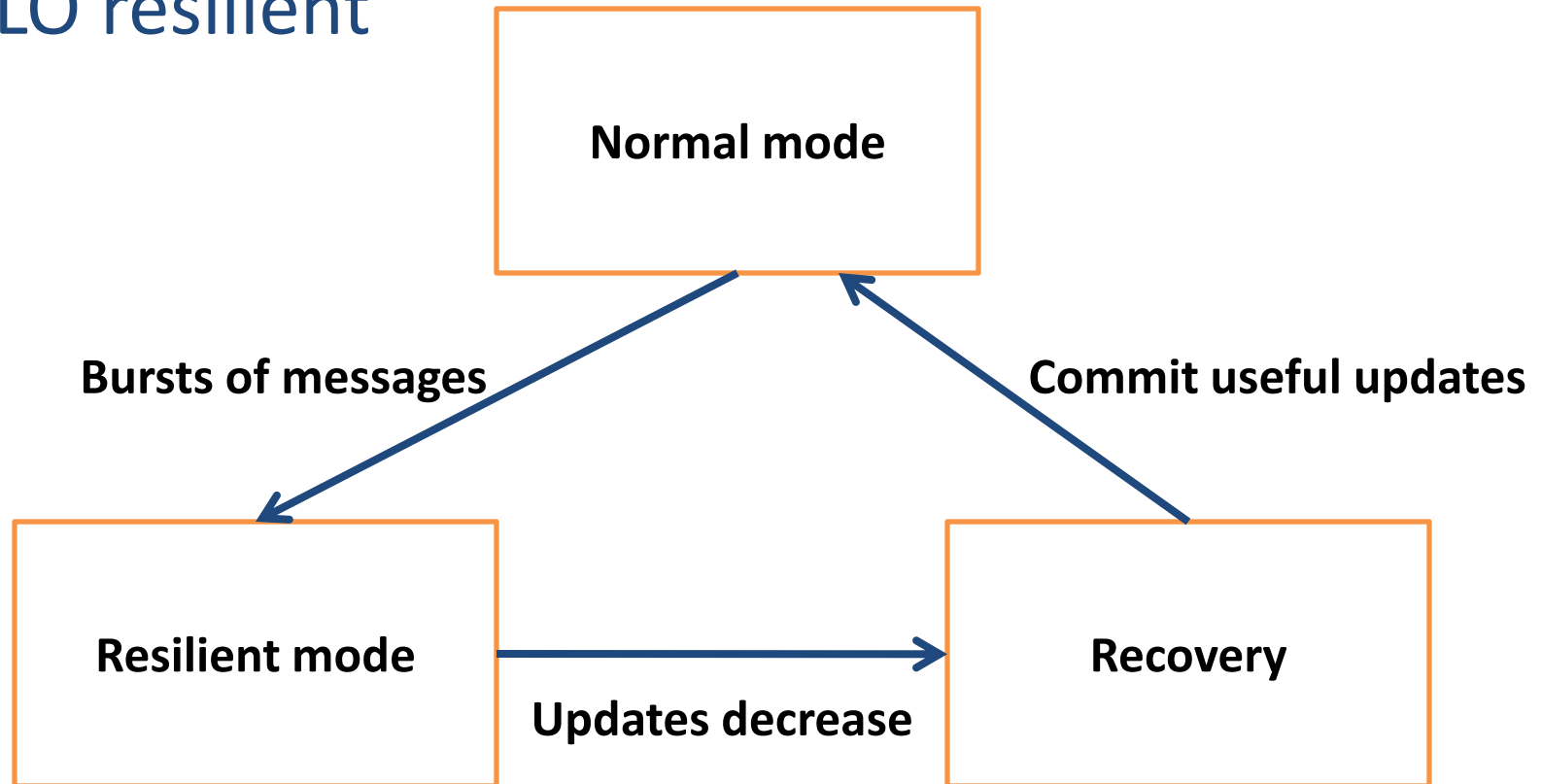
- Execution in parallel
  - Still getting delays

# Blue Waters

- First problem
  - Protect ELSA from junk messages
  - HELO resilient

# Blue Waters

- HELO resilient



# Blue Waters

- Failure identification
  - Templates identified by NCSA
  - Cray records events of interest
    - Cross correlate between each other and with templates
  - 30% failures - no template match
    - DIMM errors do not correlate with any
    - Represent 25% of failure occurrences in April 2013

# Blue Waters

- Correlations

- Offline: Decreasing the execution time

- Hierarchical algorithm (fast less reliable and slower more accurate)
    - Focus on the identified problems

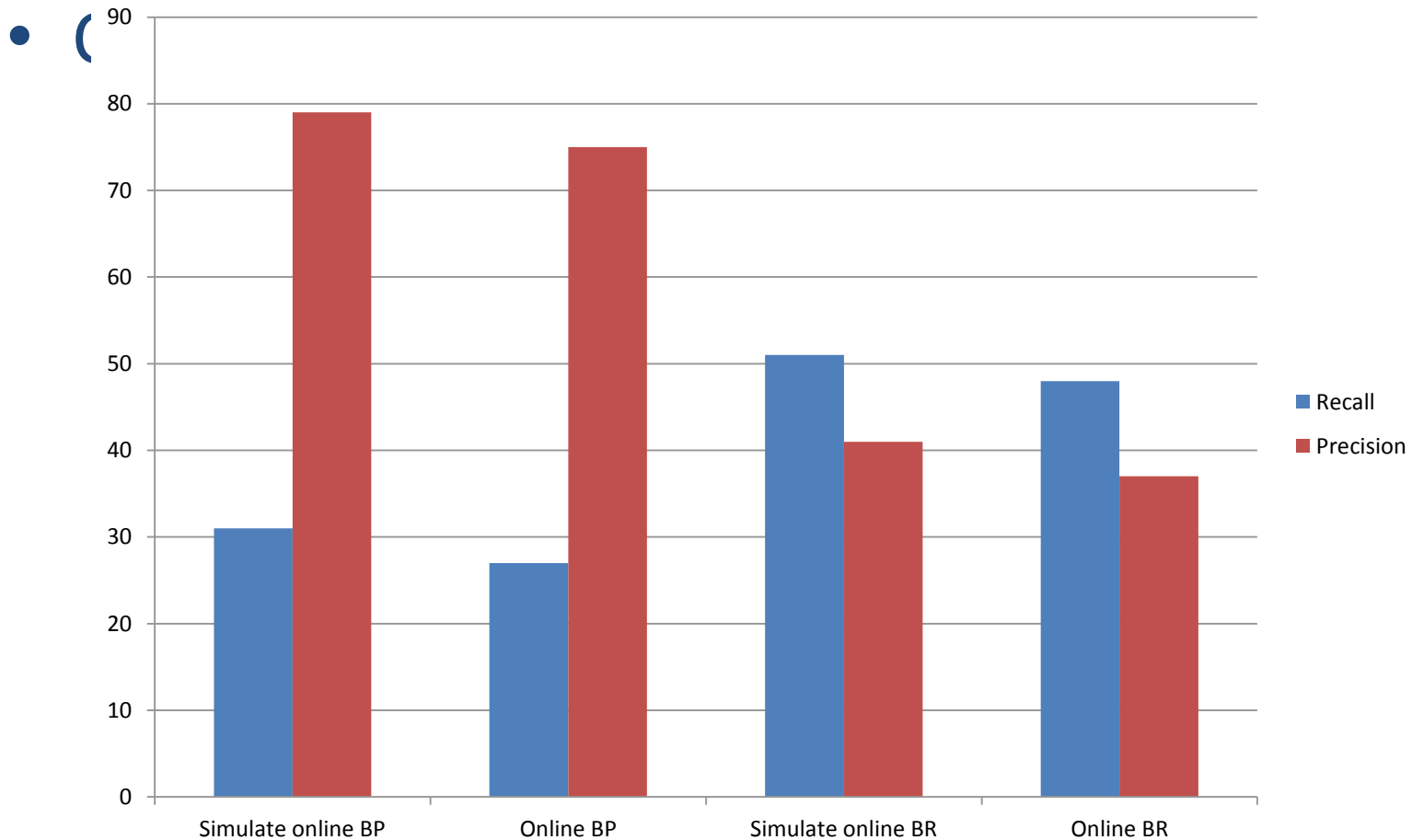
- Online: Only 42% of the type of failures

- Are part of any correlation that gives a delay greater than 10 seconds

- 13.5% of the failures affect more than one node

- Random set of locations

# Blue Waters



r  
le

# Conclusion

- What we want to achieve
  - Online failure prediction
  - Seems feasible on small systems
- Challenges with Blue Waters
  - Large execution time (in progress)
  - Sensible to log perturbations (done)
  - Some anomalies do not present any symptoms
  - Failures are not part of any correlation

# Future work



- Difference between small systems and Blue Waters
  - Anomaly detection
  - Correlation extraction
- Increase the current results by a combination
  - Different predictors



# Collaboration directions



## 1) Combining prediction with other solutions

- Collaboration with UIUC / INRIA

## 2) Using ELSA for root cause analysis

- Collaboration with NCSA / ANL (also Sandia)

## 3) Understanding failures in HPC: precursor detectors

- Collaboration with NCSA / ANL

# Additional Q&A

Thank you

Ana Gainaru  
againaru@illinois.edu