

Improving the Dependability of Grids via Short-Term Failure Predictions

Artur Andrzejak, Demetrios Zeinalipour-Yazti, Marios D. Dikaiakos



Motivation and Introduction

Reliability of Grids

- } Grids like EGEE offer sufficient capacity for even most challenging large-scale computational experiments
- } However, **Grids have notoriously low reliability:**
 - } Data processing challenges of the **WISDOM** project (2005) have shown that **only 32% (FlexX) and 57% (Autodock) of the jobs completed with "OK" status**
 - } A nine-month long study found that **only 48% of jobs submitted in South-Eastern-Europe completed successfully (*)**

(*) **Analyzing the Workload of the South-East Federation of the EGEE.** G. DaCosta, M.D. Dikaiakos, S. Orlando. *Proceedings MASCOTS 2007.*
Harvesting Large-Scale Grids for Software Resources, A. Katsifodimos, G. Pallis, M. D. Dikaiakos, *Proceedings of CCGrid 2009.*

Detecting and Managing Failures

- } Detecting and managing failures is an important step to make Grids reliable
- } This is **an extremely complex task** that relies on
 - } over-provisioning of resources
 - } ad-hoc monitoring
 - } Sys.admin & user intervention
- } Unique characteristics of Grids make it difficult to use ideas from cluster computing, Internet systems, and software systems

Why is Detecting Failures in Grids Hard?

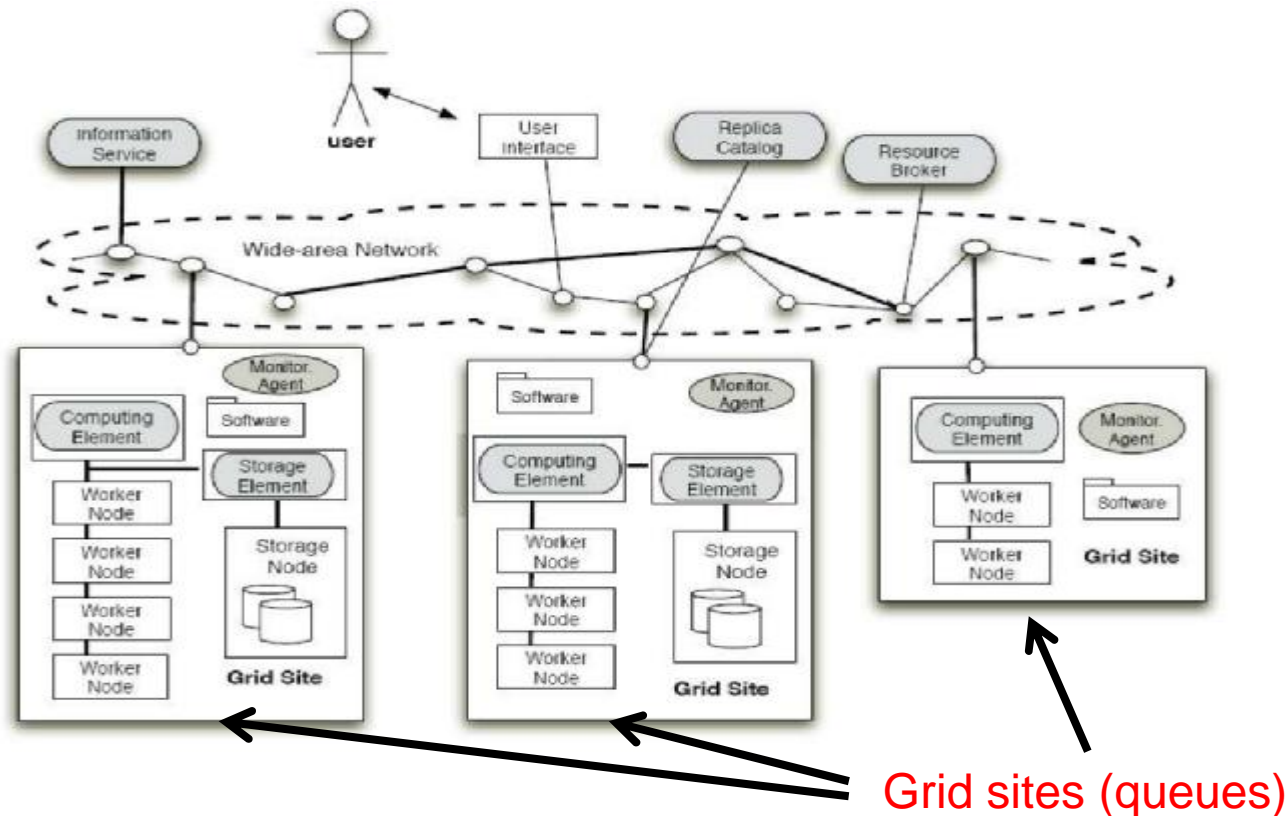
- } Lack of central administration makes it **difficult to access the remote sites** in order to monitor failures
- } Heterogeneity and legacy **impede integration of failure feedback mechanisms** in the application logic
- } Huge system size make it **difficult to acquire and analyze failure feedback data at a fine granularity**
- } It is more efficient to identify the overall state of the system and to exclude potentially unreliable sites than to identify reasons for individual failures

Failure Management in Grids: The Case of the EGEE Infrastructure.

K. Neocleous, M.D. Dikaiakos, P. Fragopoulou and E.P. Markatos, *Parallel Processing Letters*, Vol. 17, Issue 4, World Scientific, pp 391-410, December 2007

Short-Term Prediction of Site Failures

} In our approach **we predict queue (site) failures** on short-term time scale by deploying (off-the-shelf) machine learning algorithms



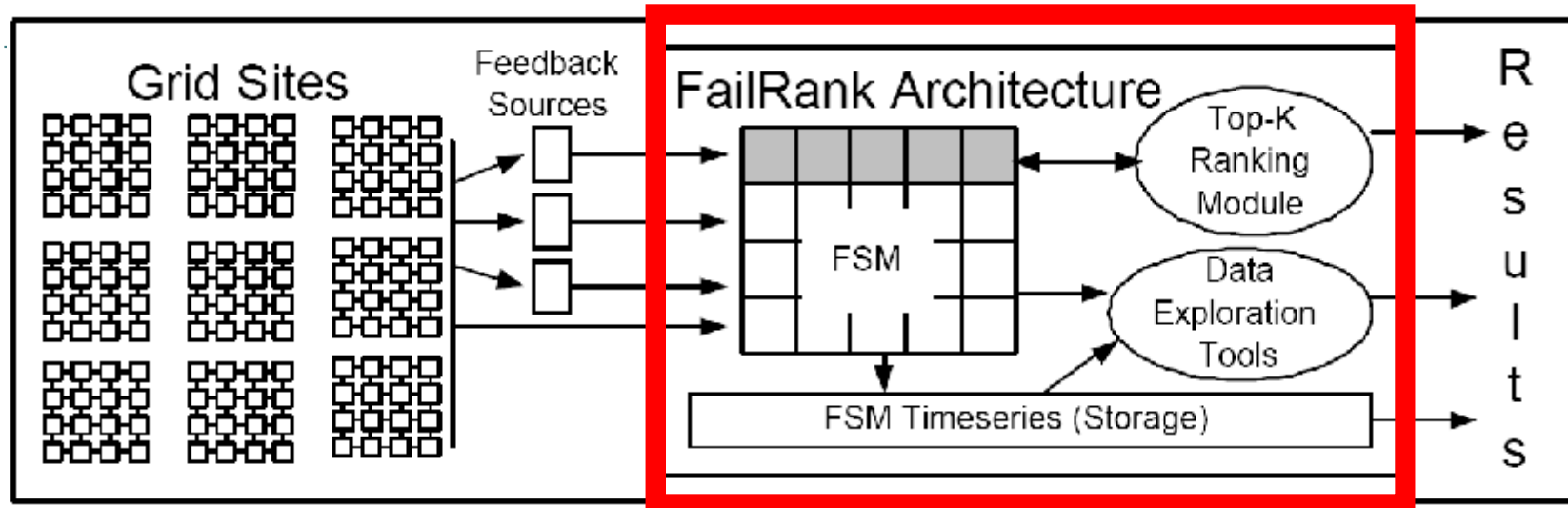
Previous Work

- } In previous work – the **FailRank** system – we have used **linear models of monitoring data**
 - } they continuously ranked K sites with the highest potential to failure
- } In this study we apply **individual models per queue** and a more sophisticated approach, including
 - } statistical selection of most meaningful sources
 - } **non-linear classification algorithms** from machine learning

"**Metadata Ranking and Pruning for Failure Detection in Grids**", D. Zeinalipour-Yazti, H. Papadakis, C. Georgiou, M.D. Dikaiakos, *Parallel Processing Letters, Special Issue on Grid Architectural Issues: Scalability, Dependability, Adaptability*, Sept. 2008.

"**Identifying Failures in Grids through Monitoring and Ranking.**" Demetrios Zeinalipour-Yazti, Kyriakos Neocleous, Chryssis Georgiou, and Marios D. Dikaiakos, in the *Proceedings of the Seventh IEEE International Symposium on Networking Computing and Applications, NCA 2008*.

FailRank Architecture



- } **FailShot Matrix (FSM):** A Snapshot of all failure-related parameters at a given timestamp.
- } **Top-K Ranking Module:** Efficiently finds the K sites with the highest potential to feature a failure by utilizing FSM.
- } **Data Exploration Tools:** Offline tools used for exploratory data analysis, learning and prediction by utilizing FSM.



Focus on Prediction Accuracy

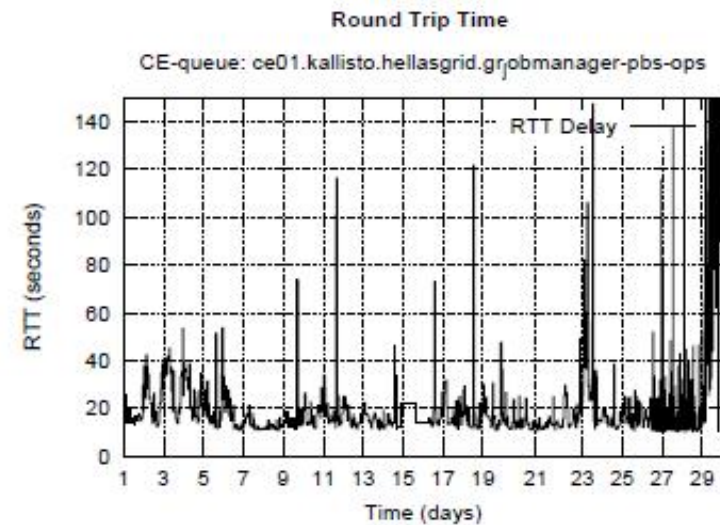
- } We **focus on** several essential questions related to **prediction accuracy**:
 - } *How many sources are necessary for high prediction accuracy?*
 - } *Which of the sources yield the highest predictive information?*
 - } *How accurately can we predict the failure of a given Grid site X minutes ahead of time?*

- } Evaluation on a 30-day trace from 197 EGEE queues shows that **prediction accuracy is highly dependent** on:
 - } the selected queue
 - } the type of failure
 - } the preprocessing and
 - } the choice of input variables

Data and Modeling Methodology

Input Data and FailBase Repository

- } Our study uses data from our **FailBase Repository**
 - } characterizes the EGEE Grid in respect to failures between 16/3/2007 and 17/4/2007
 - } maintains information for 2,565 Computing Element (CE) **queues** (sites accepting computing jobs)
- } For our study **we use a subset of 197 queues** with most types of monitoring data
- } For *each queue* data is a sequence of pairs (timestamp, **attribute vector**)
 - } Each attribute vector consists of 40 measurements from to various sensors and tests
 - } **Sampled every 1 minute**



Exemplary attribute (RTT) over time

Types of Input Data

- } **A. *Information Index Queries (BDII)***: 11 attributes from LDAP queries
 - } e.g. number of free CPUs; max. number of running and waiting jobs
- } **B. *Grid Statistics (GStat)***: processed data from the monitoring web site of Academia Sinica
 - } e.g. geographical region of site; available storage space
- } **C. *Network Statistics (SmokePing)***: Data of the gPing database from ICS-FORTH
 - } average round-trip-time (RTT); the packet loss rate
- } **D. *Service Availability Monitoring (SAM)***: 14 attributes derived from raw html published by the CE sites
 - } e.g. the version number of the middleware; results of various replica manager tests; results from test job submissions

Predictive Models

- } Our prediction methods are **model-based**
 - } A model in this sense is a **function f** mapping vectors of sensor values to an output (queue healthy (0) or not (1))
- } We use as models **classification algorithms**
 - } Classifiers "learn" the relationship between input data and the output ("**class value**") based on historical examples
 - } They are well-established in data mining and have been perfected over time
- } We deploy several common classifiers
 - } C4.5 (decision tree), AdaBoost, Naive Bayes, LS

Learning the Model

- To predict, we need **to learn the relationship** between inputs (A, B) @ "now" and the value of our model $f @ (\text{now} + T)$
- First stage (**training, model fitting**):
 - Supply training data consisting of triples $[A@x, B@x, f@(x+T)]$ sampled at different times x
 - Then learn a function which captures this relation
- Second stage (**prediction**): supply (A,B) and compute $f @(x+T)$

↖ @ = "at time"

	Metric A	Metric B	f
Example 1	60	1000	[30-33]
...
Example k	1.4	10^6	[3-6]
Unknown Sample	30	50000	?

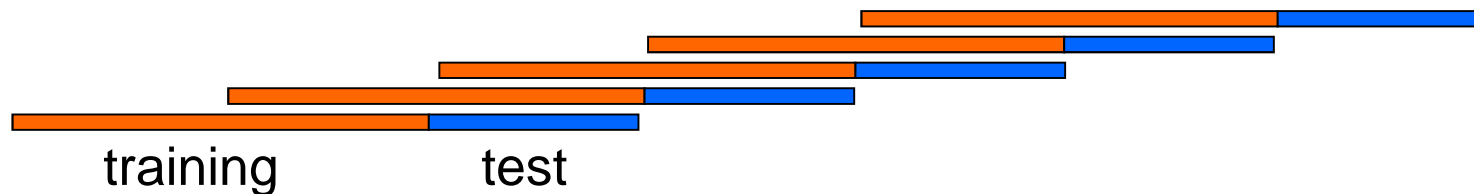
} 1. fit model
└ 2. predict

Evaluation Metrics

- } To quantify prediction errors we use
 - } **Recall** = probability that a (randomly selected) failure is indeed predicted
 - } **Precision** = probability that a (randomly selected) failure prediction indicated a true failure
- } These metrics are then **averaged over all 197 queues** for most diagrams

Model Updates

- } Models are periodically updated to ensure adaptability to profile changes
 - } How? Train model on the orange part and test on the blue part, then advance by the blue part etc.



- } The used values are:
 - } **training interval: 15 days** (21600 of 1-minute samples)
 - } **update interval = test interval = 10 days** (14400 samples)
 - } why? – will be shown later

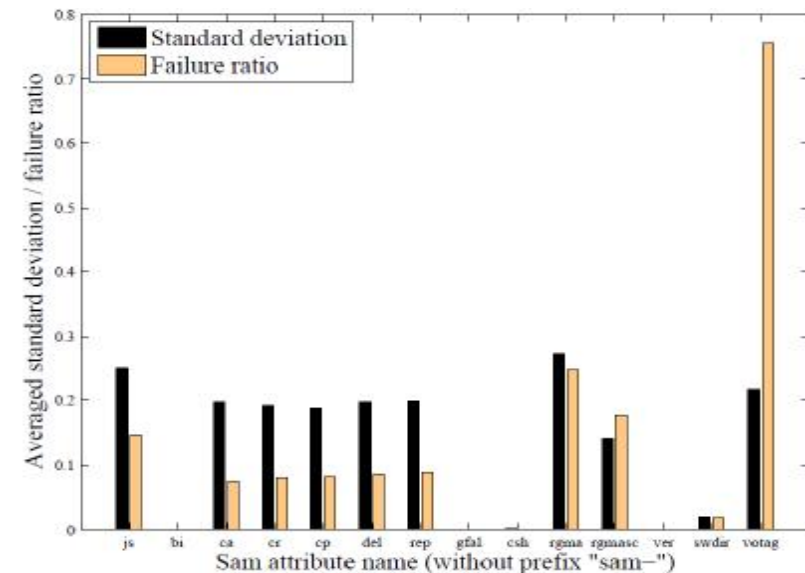
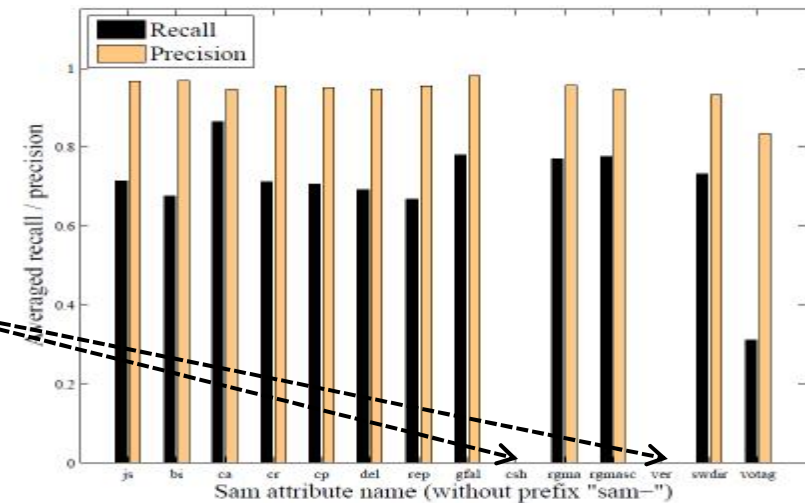
Experimental Results

Identifying Failure Indicators

- } Unfortunately, we do not have any additional data whether jobs on a site have failed or not
- } As a substitute, we used as **failure indicators** two metrics from the Service Availability Monitoring (SAM) measurements (group D):
 - } **sam-js**: a test that submits a simple job for execution to the Grid and then seeks to retrieve that job's output from the UI
 - } **sam-rgma**: R-GMA makes all Grid monitoring data appear like one large DB; this test insert a tuple and run a query for that tuple
- } *Values (0/1) of each of these 2 metrics are assumed to mean "queue failed" or "queue healthy"*

Why sam-js and sam-rgma?

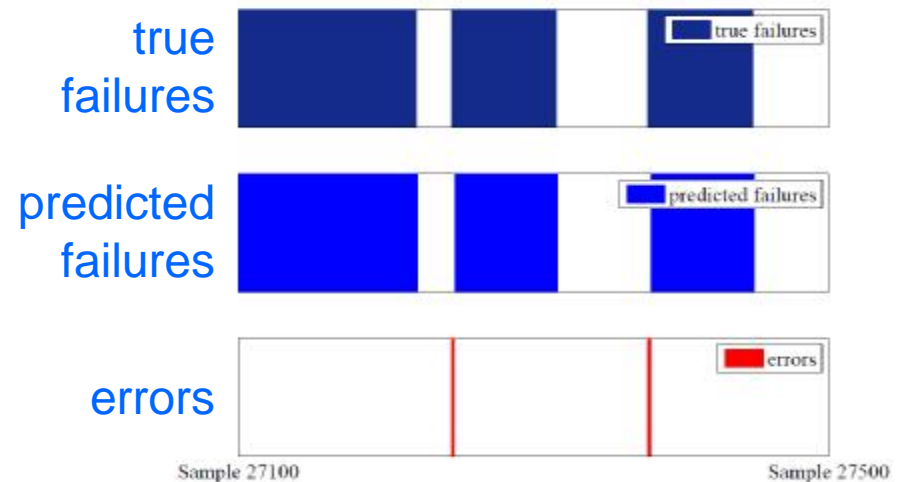
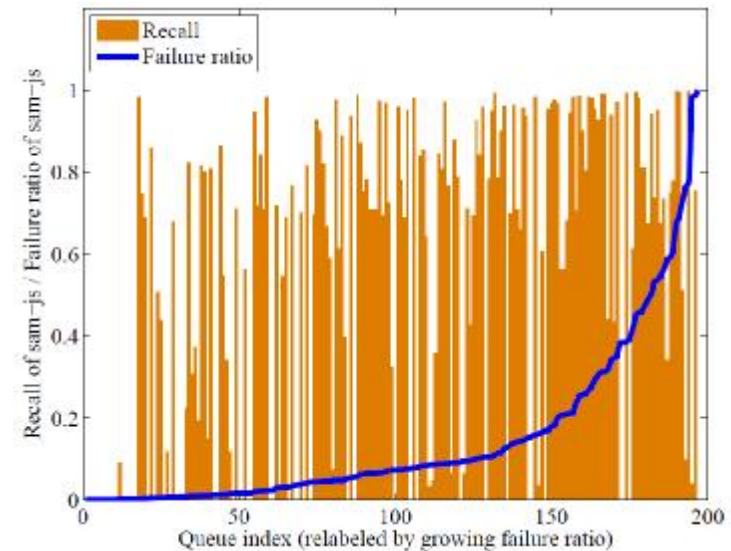
- } First: we computed averaged recall / precision for all 14 SAM (group D) attributes
- } This eliminated only two of them
- } We then looked *per attribute* at:
 - } **standard deviation** - more changes = more information
 - } **failure ratio** = (#all samples indicating a failure) / (# all samples)
 - } low FR = not enough "bad cases" to train a predictor
- } From the remainder ones **we selected these 2 by importance of representing failures**



Data Characteristics

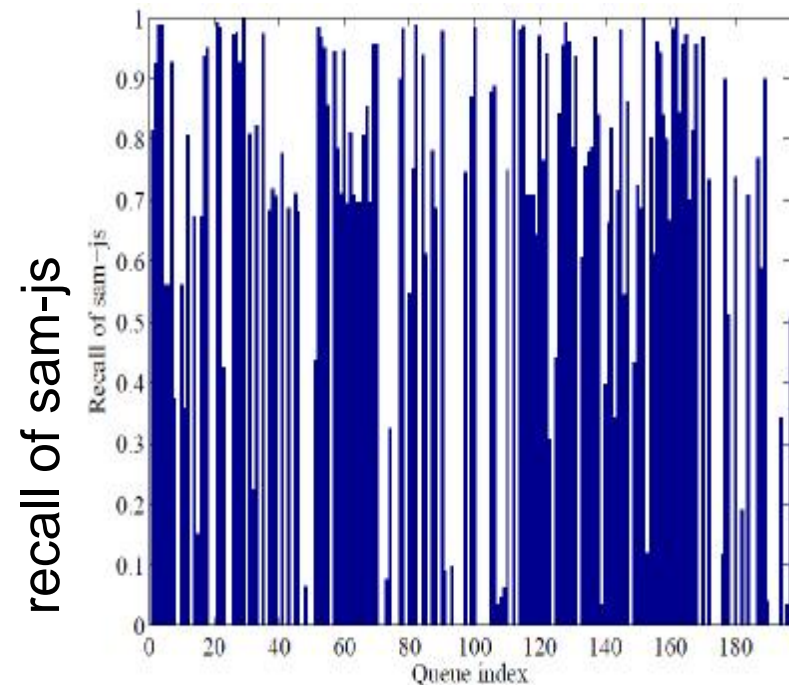
- } A. Is there a relationship between *failure ratio (FR)* and accuracy?
 - } $FR = (\# \text{all samples indicating a failure}) / (\# \text{all samples})$
- } Plot: recall of sam-js (bars) sorted by FR or sam-js (line)
- } **No!** => **Models are "non-trivial"**

- } B. What are the *failure patterns* in our data?
- } Typically, the failure state does not change frequently (long "runs" of failures / non-failures)
- } Prediction errors occur frequently right after the change of failure state



Are Individual Models (per Queue) Useful?

- } We have created separate model (trained classifier) per queue
- } This is a lot of effort – is it useful?
- } It turns out that prediction **accuracy varies hugely between queues!**
- } Lessons:
 - } "Aggregated models" of reliability (i.e. one model for many queues) can be severely inappropriate
 - } **Scheduling decisions should take into account confidence of the model per queue**
 - } How likely is to predict a failure for this queue?
 - } If confidence is low, increase redundancy / overprovision for this queue *preemptively*

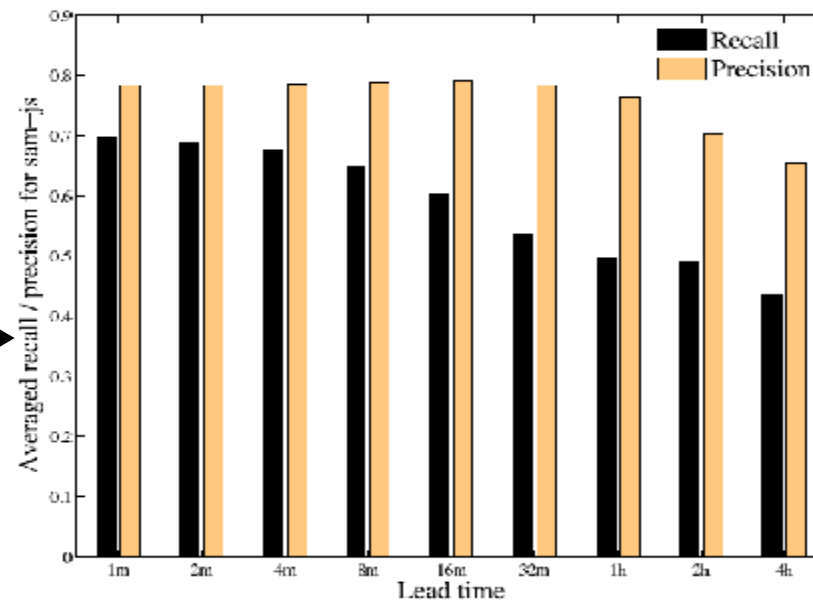


queue index – 0 to 196

Lead Time vs. Accuracy

- } How much into future can we predict?
 - } we set the **lead time** to **15 minutes**
 - } lead times of 1-8 minutes were slightly more accurate
 - } but not very useful – might not give enough time to react
 - } lead times above 30 minutes yielded larger errors

Averaged recall & precision for sam-js

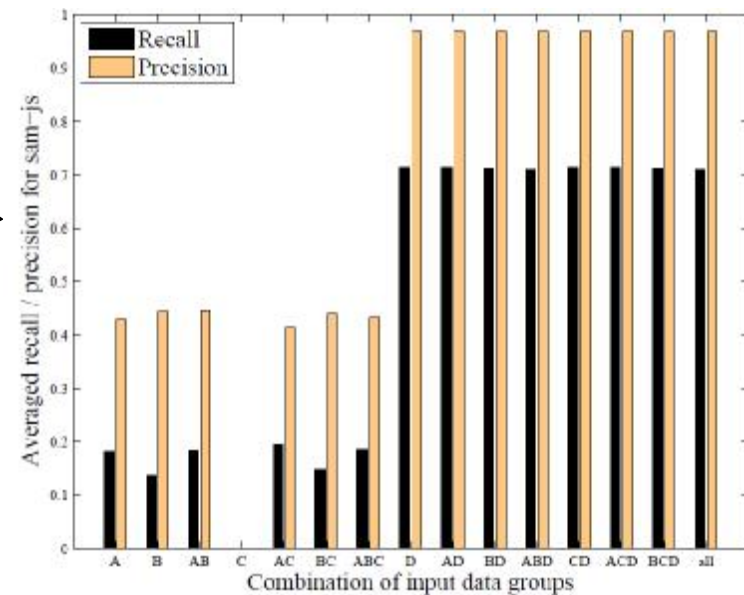


lead time (in minutes) 1 4 16 60 4*60

Most Relevant Types of Input Metrics

- } Which of the input types (A, B, C, D) provide most predictive information?
- } We tested all input combinations A, B,..., AB, AC,..., ABCD
- } Group D (SAM = functional tests) is most relevant
 - } In fact, groups A, B, C do not carry any additional information

Averaged recall & precision for sam-js

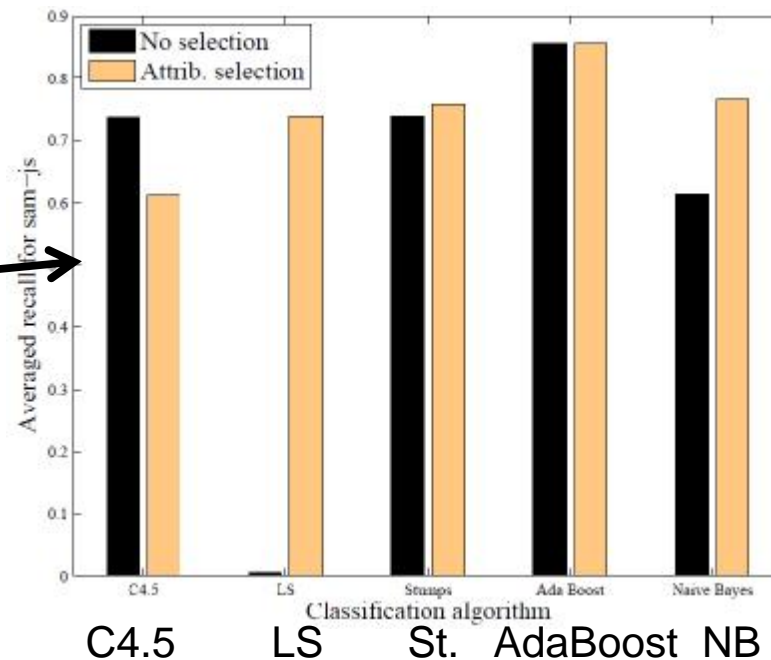


Group combination → A B AB C AC BC ABC D ... BCD.all

Classifier Type & Attribute Selection

- } Are some classifiers more accurate than others?
 - } Except for the least sophisticated algorithm (LS = linear perceptron, a hyperplane in R^d) accuracy is comparable
- } How much attribute selection matters
 - } Mixed results: for LS & Naïve Bayes improvement, for C4.5 (decision tree) deterioration

Averaged recall for sam-js
(no selection /
with attribute selection)



Conclusions

- } Short-term prediction of failures in Grid queues can yield high accuracy (precision / recall)
- } However, this **accuracy varies hugely among queues**
 - } Individual queue modeling is essential
- } **Some metrics** (like *Service Availability Monitoring (SAM)*) **are more informative** than all others together
 - } Consider this for "economical" metric collection
- } Sophisticated classification algorithms yield comparable accuracy

Future work

- } Direct comparison with FailRank (linear models)
- } Scheduling strategies with consideration of model confidence

Additional Slides

Why is Detecting Failures in Grids Hard?

- } Lack of central administration makes it **difficult to access the remote sites** in order to monitor failures
- } Heterogeneity and legacy **impede integration of failure feedback mechanisms** in the application logic
- } Huge system size make it **difficult to acquire and analyze failure feedback data at a fine granularity**
- } It is more efficient to identify the overall state of the system and to exclude potentially unreliable sites than to identify reasons for individual failures

Exploiting Generic Feedback Sources

- } Instead of using application-specific feedback data, **we exploit a set of generic feedback sources**
 - } representative low-level measurements (SmokePing)
 - } websites, e.g. Grid Statistics (GStat)
 - } functional tests and benchmarks
- } Such predictions can be used for deciding where to submit new jobs and help operators to take preventive measures

Classifiers Explained Visually

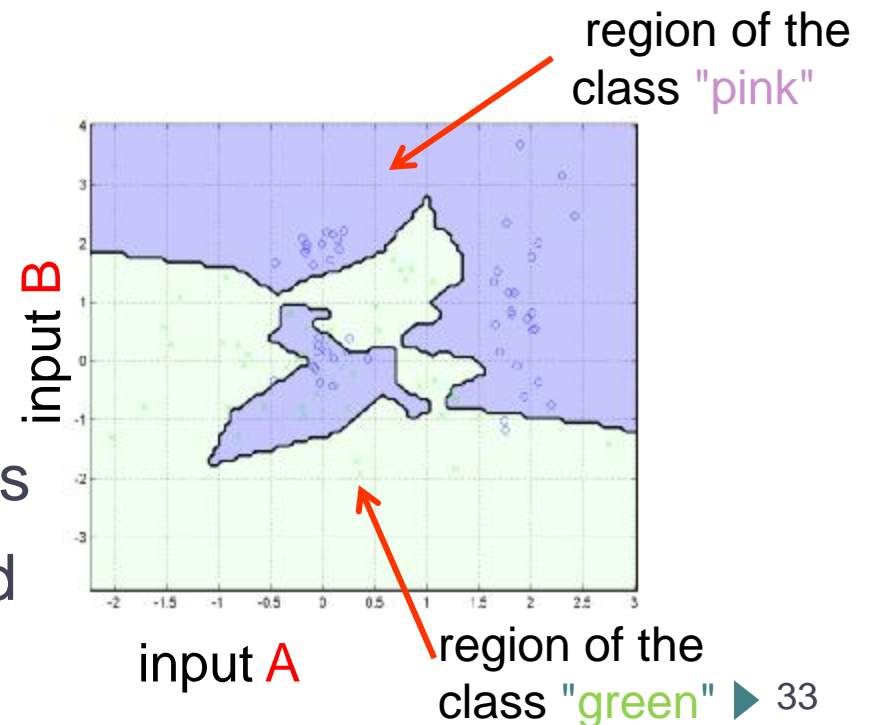
- Assume that you have **two metrics**, and want to use them for predicting some (discrete) value - a **class**
 - Interpret inputs as coordinates of points in the plane
- Then **training data = multicolored points in R^2**
 - color corresponds to a class (here: healthy or no)

- **Training:** *finding a suitable subdivision of the plane*

- **model** = a compact representation of a colored subdivision

- **Prediction:** given a new sample, find its color = class

- We have 40 metrics instead of 2 (R^{40}), but same idea

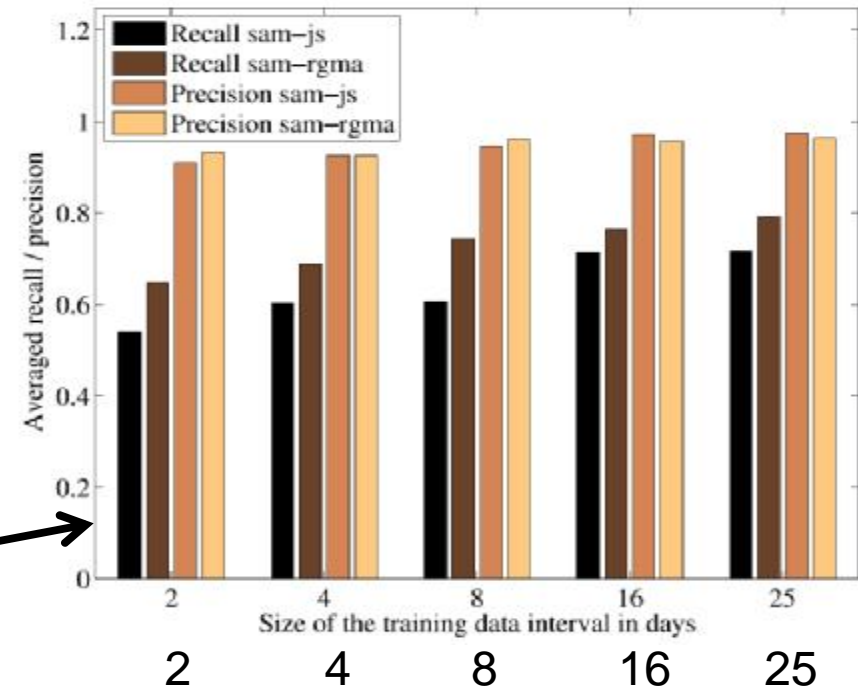


Attribute Selection

- } Initially, we do not know which of the 40 metrics (= attributes) contain most predictive information
- } Keeping all create some serious problems
 - } Overfitting
 - } Inefficiency: memory "explodes" at training phase
 - } We don't learn which metrics are really relevant
- } Therefore we use **attribute selection**
 - } Learn and evaluate "probe models" on training data with various subsets of attributes
 - } Then use attribute sets with lowest errors
 - } *For specialists*: we use forward or backward branch-and-bound selection with C4.5 (decision tree)

Training Data Size

- } How much training data (# samples) is needed for accurate models?
- } In general, the less the better
 - } Higher adaptability to changes, less "waiting time" until first results
 - } But too little data decreases accuracy
- } **Training interval of 15 days** turned out optimal
- } Test interval = Model update interval was irrelevant



Averaged recall & precision
vs. training time (days)