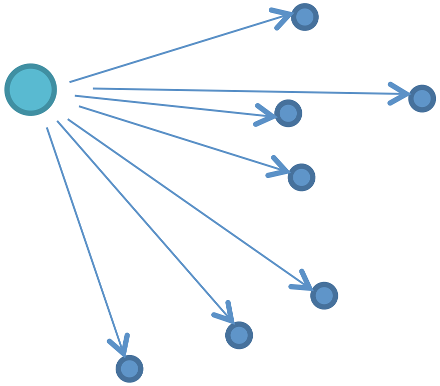
Latent Fault Detection With Unbalanced Workloads

Moshe Gabel

With Assaf Schuster Danny Keren,
Kento Sato @ LLNL Satoshi Matsuoka @ TITECH

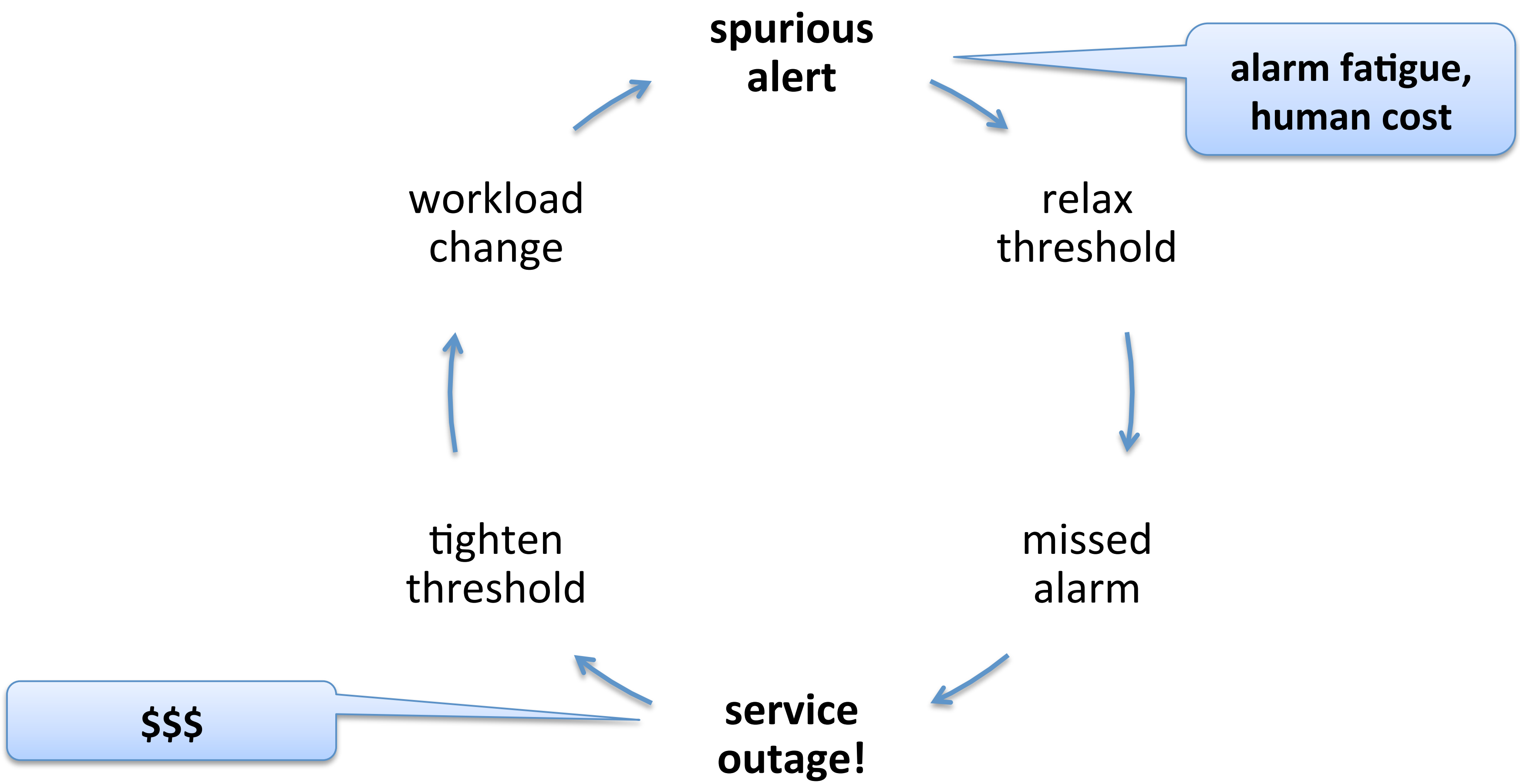
Submitted to EPForDM 2015



Background

LATENT FAULT DETECTION

The Problem With Predefined Rules



Flexible Latent Fault Detection

- Find ***latent faults*:** machines with problems “under the radar”.
- Latent faults precede > 20% of failures **days in advance**.
- Outlier detection on performance counter logs.

GOOD – EASY, FLEXIBLE, PRACTICAL:

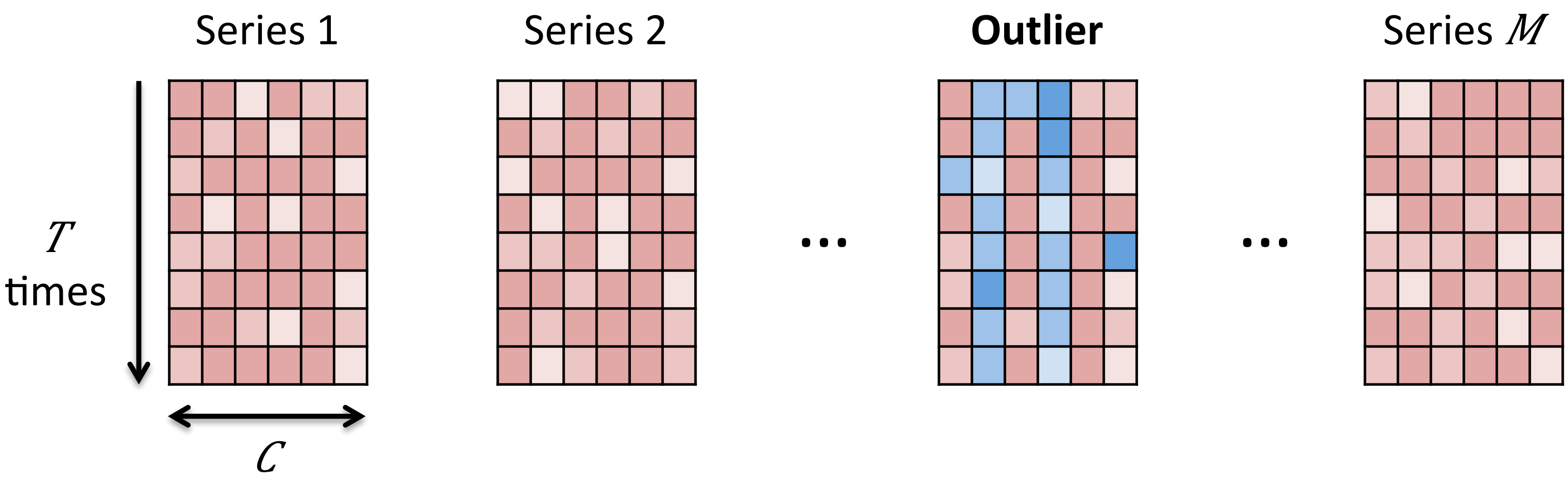
- **Predict failures** up to 14 days in advance with high precision.
- No tuning, specialized knowledge, or labeled examples.

PROBLEMS – CENTRALIZATION, LOAD BALANCING:

- **Large data:** communicating and processing machine metrics.
- Only for **load-balanced** services.

TASK: Find Outliers

- Given M multivariate time series of C measurements...
- Machines in scale-out, load balance service.

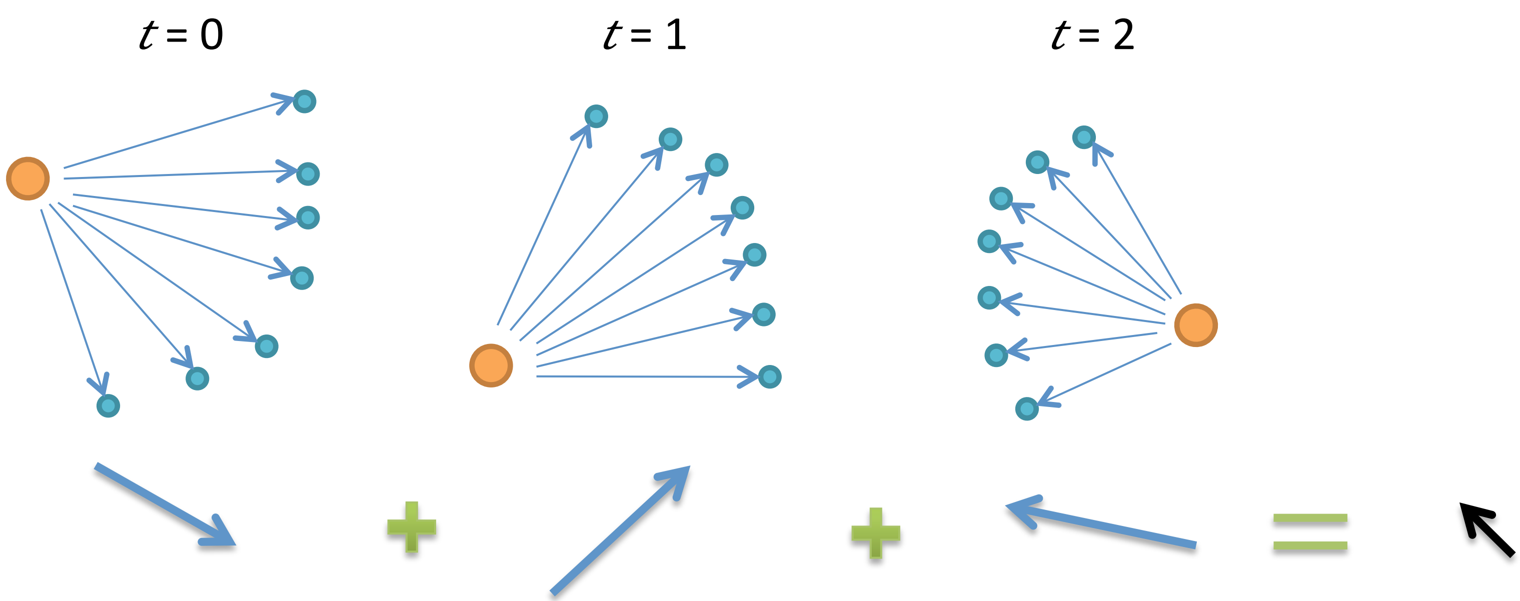


- Task: find outliers – series with "bad" behavior.
  - Example: machine with HW/SW error

IDEA: Wisdom of the Crowds

▶ Exploit suitable **homogeneity** assumptions:

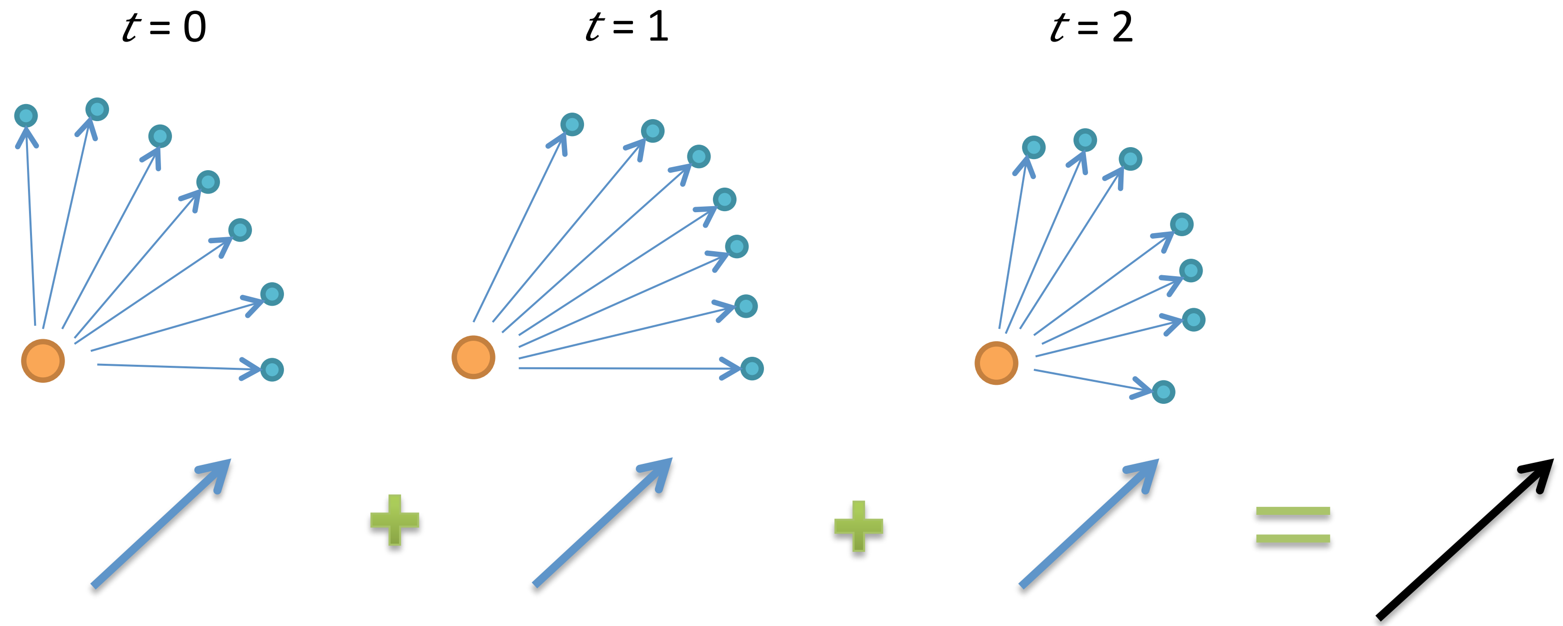
Similar processes (machines) will exhibit **similar behavior**.



Outliers Are Different

► Outliers come from different processes – break homogeneity:

Outliers (faulty machines) are **consistently different**.



Sign Test: Is Machine i an Outlier?

- At each time: average direction from i's vector to others.
- Add the average directions across T times; compare lengths.
- Compute probability $p_i = \Pr[$ series i not outlier $]$.
  - Via concentration bounds or something else.
- p_i too low → series i is an outlier.

Workload	Centralized	Distributed
Balanced		
Unbalanced	✓	

Submitted to EPForDM 2015

With Kento Sato @ LLNL, Satoshi Matsuoka @ TITECH

LATENT FAULT DETECTION WITH UNBALANCED WORKLOADS

Detect Latent Faults In More Settings

Go beyond load-balanced, scale out web services:

- Unbalanced cloud workloads
  - Statically-balanced key-value stores
- Parallel computation clusters
  - Hadoop
- Supercomputers
  - TSUBAME2

Central Assumptions

- Homogenous machines
  - Common for logistical reasons
- Majority of machines are OK
  - Otherwise systems don't work
- Dynamic load balancing
  - Hadoop and similar have unbalanced workloads

> Y. Kwon, K. Ren, M. Balazinska, and B. Howe.
> Managing skew in Hadoop. IEEE Data Eng. Bull., 2013.

  - Supercomputers: uneven work distribution

Assume Intrinsic Correlations

- Inherent dependencies exists between counter values.
  - (not necessarily linear, pairwise)
- Characterize running job – same regardless of load.
- Example: for each client request we need:
  - 10MB of memory, 3 DB transactions, 2% CPU

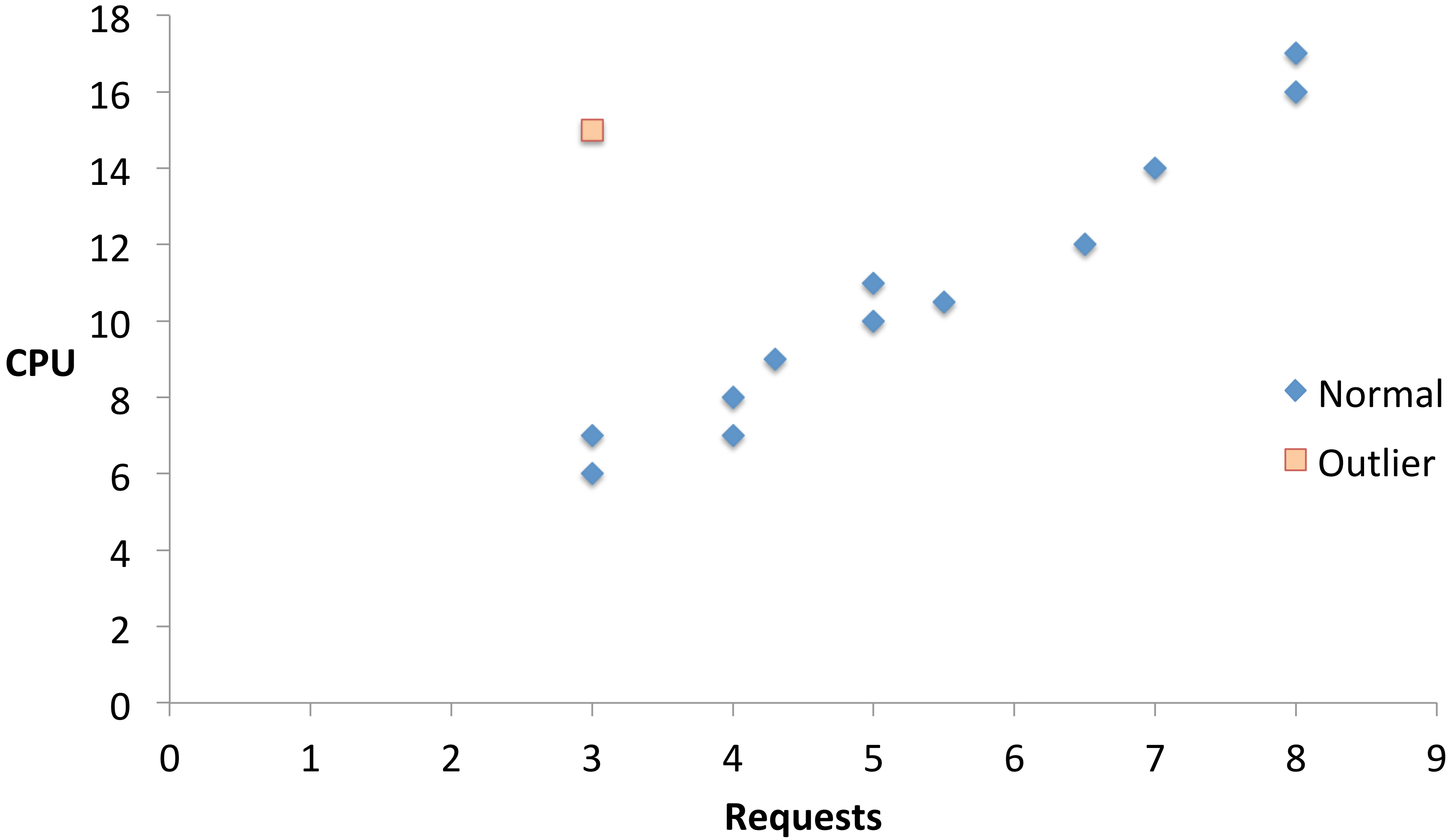
Requests	Memory	DB	CPU
3	630	9	6
5	650	15	10
4	640	12	8

Faults Break Correlations

- Established rule:
  $1/10\ memory + 1/3\ DB - 1/2\ CPU - requests - 60 = 0$
- Problems cause deviation from established relationships.
  - DB errors, memory leaks, high CPU usage…

Requests	Memory	DB	CPU
3	630	9	6
5	650	15	10
4	640	12	8
3	**740**	9	**15**
8	680	24	16

CPU Usage vs Requests



New Strategy

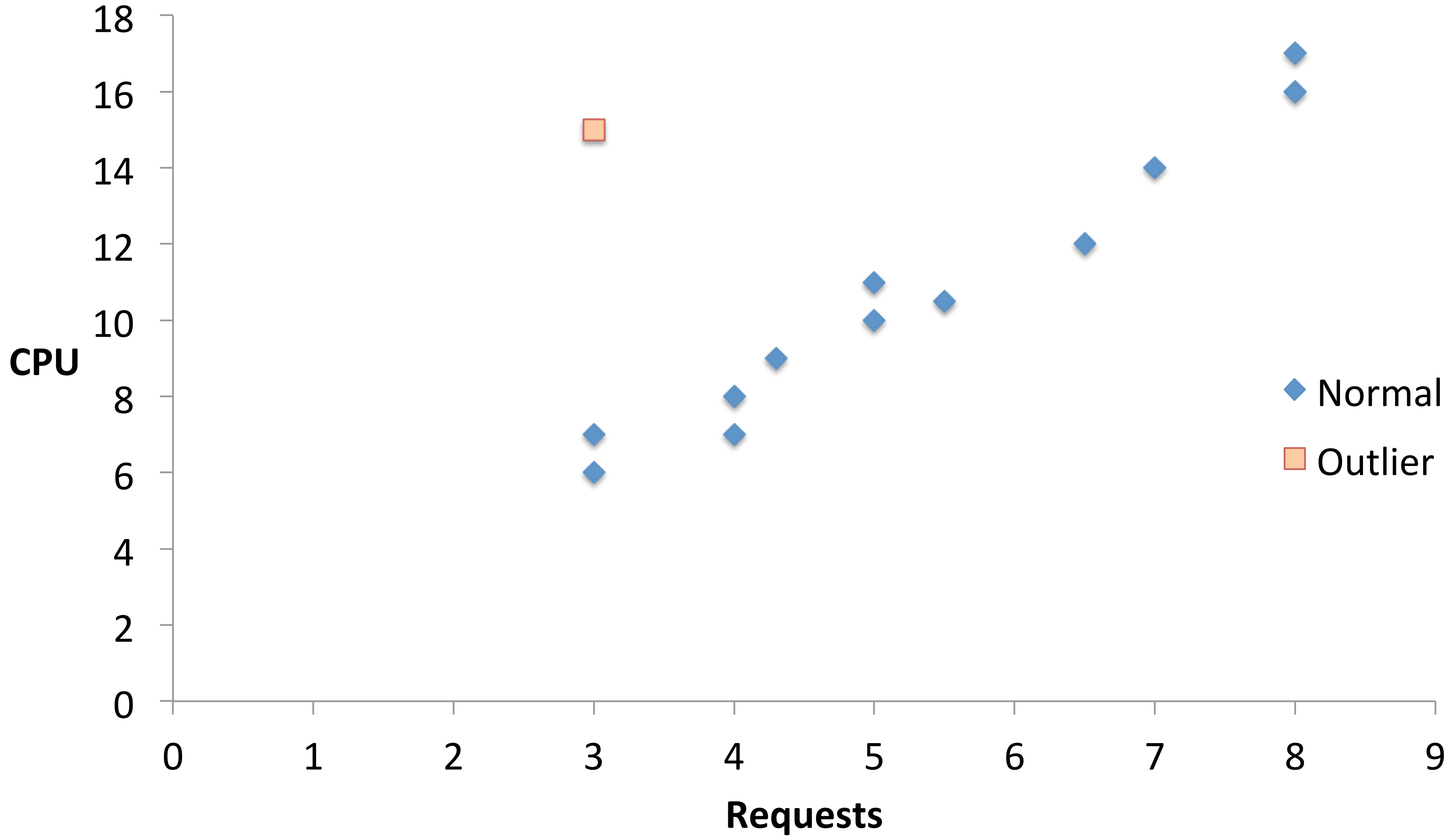
- New assumption:
  similar machines doing same work → similar correlations.
- Establish linear correlations at each point in time.
- Machine consistently breaks correlations? → latent fault.
- Limit false positives via statistics.

- J. E. Jackson and G. S. Mudholkar. Control procedures for residuals associated with principal component analysis. Technometrics, 1979.
- H. Xu, C. Caramanis, and S. Mannor. Outlier-robust PCA: The high-dimensional case. IEEE Transactions on Information Theory, 2013.
- M. Gabel, A. Schuster, R.-G. Bachrach, and N. Bjorner. Latent fault detection in large scale services. In Proc. DSN, 2012

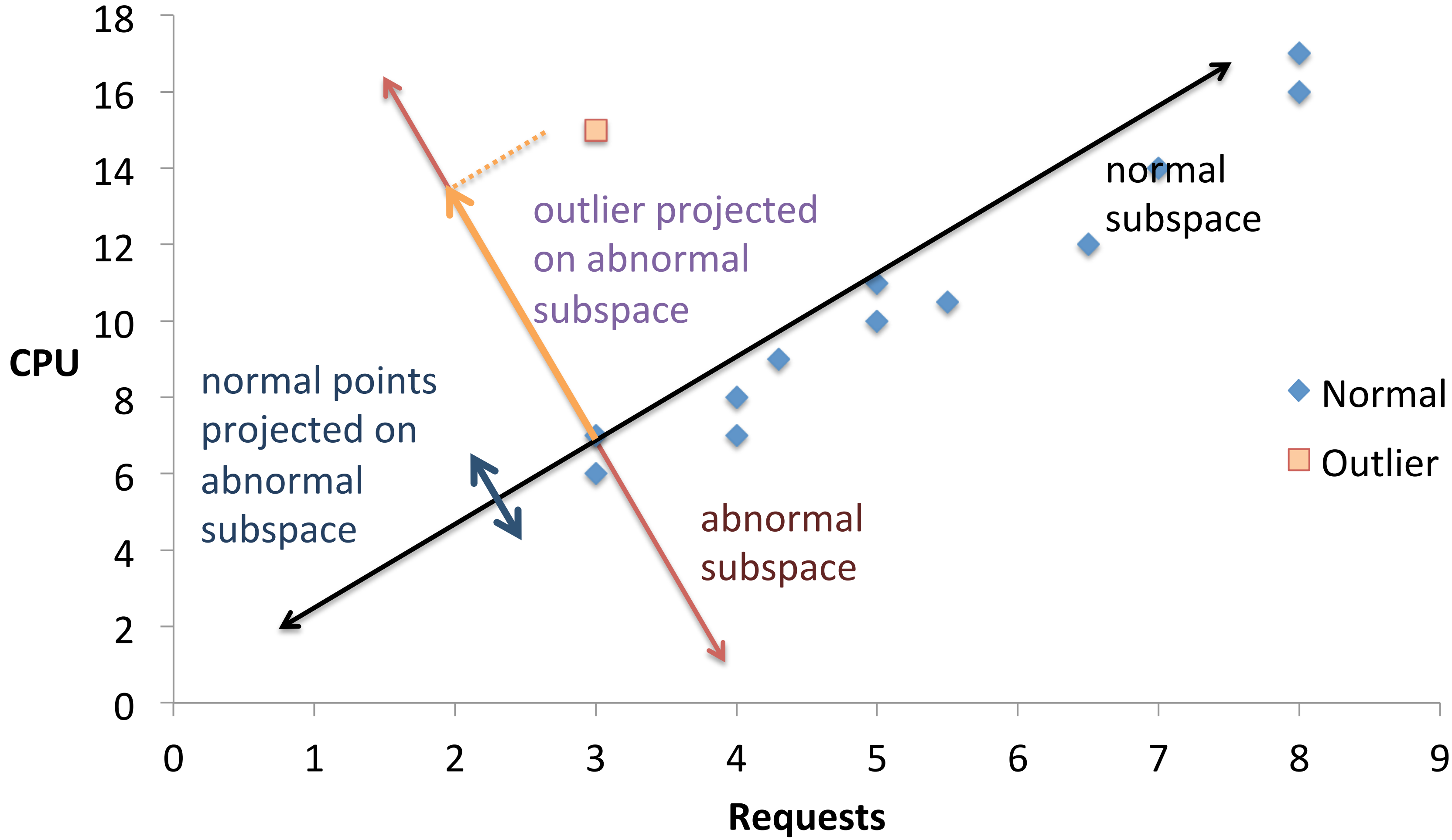
PCA Subspace Decomposition

- Counters = **normal subspace** + **abnormal subspace**.
  - Top principal components capture most variance
  - Normal subspace = top principal components = healthy correlations
  - Abnormal subspace = residual subspace
- Learn normal subspace from majority of machines.
  - Historical data unreliable or irrelevant.
- Project to abnormal subspace. Large projection? → outlier
  - Statistical guarantees: Jackson and Mudholkar 1979, Gabel et al. 2012.
- HR-PCA (Xu et al. 2013) robust to outliers, corrupted data.

Subspace Decomposition

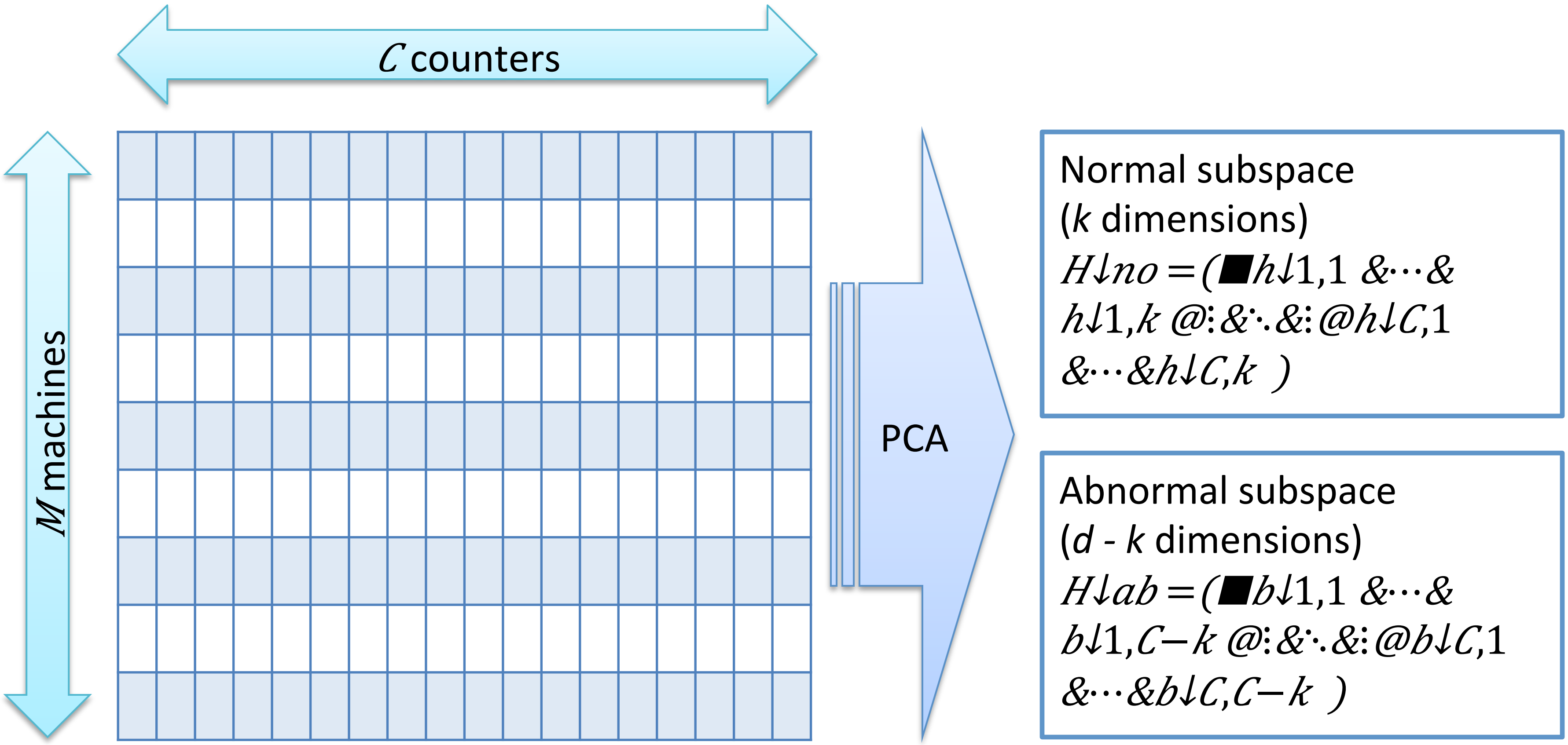


Subspace Decomposition



At Each Point of Time

- We have many machines at each point in time:



Variant 1 – Hard Threshold

- Create $M \times C$ matrix and apply PCA.
  - Standardize data to zero mean, unit variance.

- Normal subspace: $H_{no} = [v_1, v_2, \ldots, v_k]$
  - First *k* principal components that capture 95% of variance.
- Abnormal subspace: $H_{ab} = (I - H_{no} H_{no}^T)$

- Project machine data to abnormal subspace: $Q_m = \|H_{ab} x_m\|^2$

- If $Q_m > Q_\alpha$ consistently, machine is suspect.
  - Threshold Q_α from Jackson and Mudholkar 1979.

Dealing With Many Machines

- Q_α guarantees false positive rate α for testing one machine.
  - We must test M machines!

- Raise alarm only if $Q_m > Q_\alpha$ for T' **consecutive times**.

- False alarm probability decreases **exponentially** in T'.
  - *False alarm in specific machine m in T' consecutive times: $\alpha^{T'}$*
  - *False alarm in at least one machine after T' times: $1-(1-\alpha^{T'})^M$*

- Window T' that guarantees final false alarm probability p:

$$T' = \lceil \log_\alpha (1-\sqrt[M]{1-p}) \rceil$$

Variant 2 – Latent Fault Framework

- Hard threshold too strict: high Q_α in noisy data.
  - $Q_m < Q_\alpha$ even for faulty machines → missed faults.

- Statistical framework from Gabel et al. 2012:
  $S(m,x(t)) = Q_m / \|x_m\|^2 = \|H_{ab} x_m\|^2 / \|x_m\|^2$
- Integrate for each machine:
  $v_m = \frac{1}{T} \sum_t S(m,x(t))$
- Get p-value:
  $p(m) = (M+1) \exp\left(-2TM\gamma^2 / (\sqrt{M}+1)^2\right)$

Probability Bound Tightness with *M*

- Old Framework: probability **linearly weaker** with more machines:
  $p(m) = (M+1)\exp\left(-T^2\gamma^2 M/(\sqrt{M}+1)^2\right)$

higher *M* weakens bound

higher *T* tightens bound

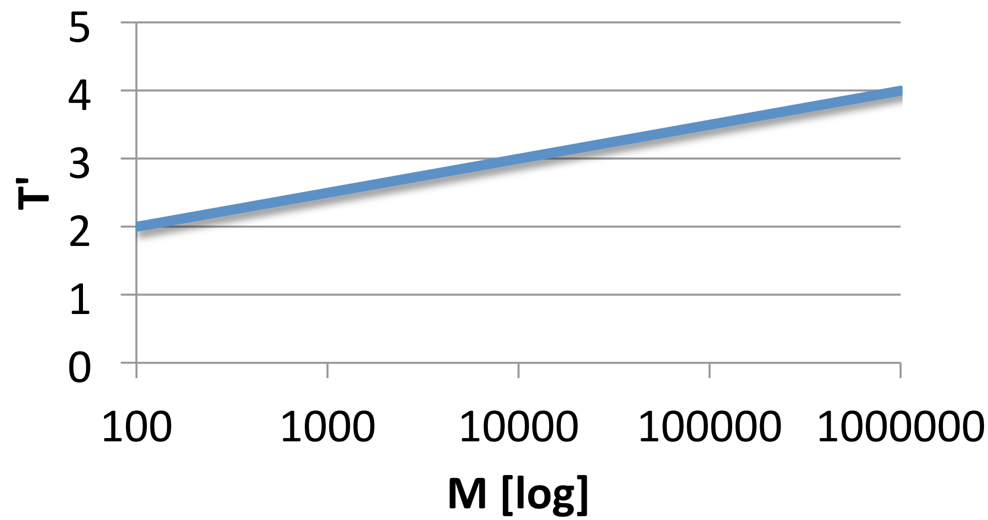
weaken bound for low *M*
no effect for high *M*

- Had to increase window size T to compensate for large M.

- New bound: false alarm probability drops **exponentially** in M:
  $p = 1 - (1 - \alpha^T)^M$

Very Small Window *T'*

- T' **logarithmic** in M.

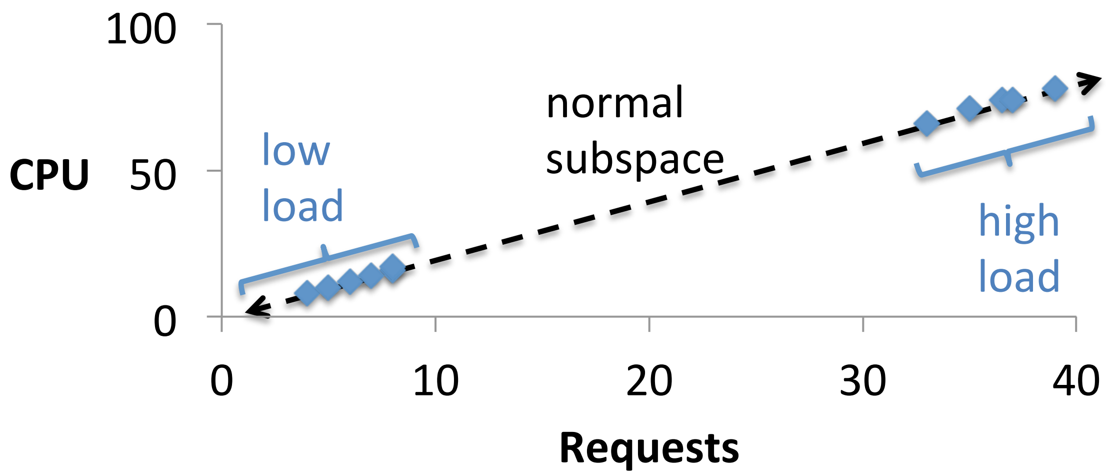


- Examples for $\alpha=0.01$, false alarm $p < 0.01$.

- $M=10000$ machines → need window size of just 3:
  $T' = \lceil \log_{0.01} (1-\sqrt[10000]{1-0.01}) \rceil = \lceil 2.99891\ldots \rceil = 3$
- For **one million** machines, need **window size = 4**:
  $T = \lceil \log_{0.01} (1-\sqrt[\mathbf{1000000}]{1-0.01}) \rceil = \lceil 3.99891\ldots \rceil = 4$

Unbalanced Workloads

- Healthy machines have same correlations.
- Normal data lies in normal subspace!



- PCA recomputed each time → robust to changes in system!

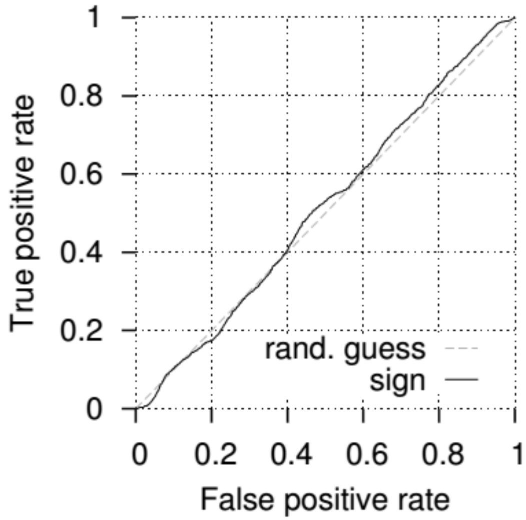
Preliminary Results on Supercomputer

- TSUBAME2 logs of one month of "jobs"
  - No scheduling info.
  - CPU and GPU load used to infer grouping.
  - At least 10 machines per job, at least 240 minutes.
- 45 common metrics, collected every 1-10 minutes.
- Compare to historical failure logs – 7 day horizon.
  - Failure probability per day: roughly 0.2%

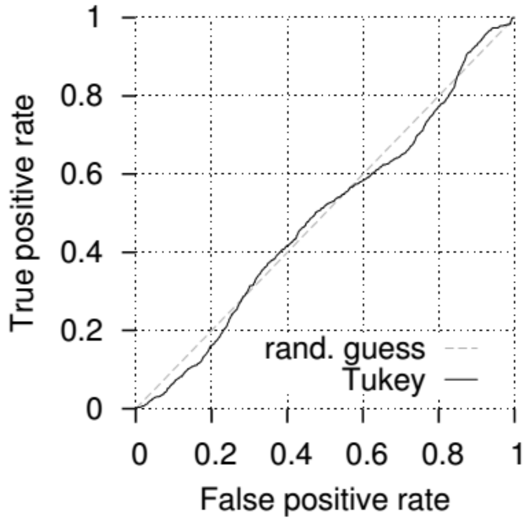
Original Latent Fault Detector

▶ Complete failure: no better than random guess.

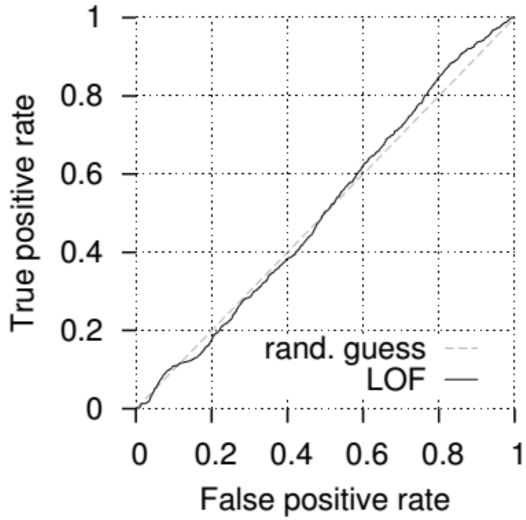
Sign test



Tukey test

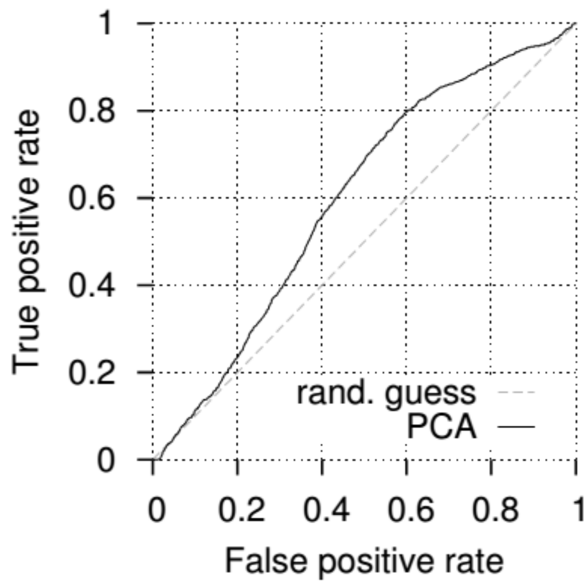


LOF test



PCA (Variant 2, "soft threshold")

- Significant improvement!
- Hard threshold variant too conservative.
  - Issued no alerts.
- Not yet practical.
  - Low FPR → low TPR.
- Ad-hoc grouping problematic.
  - excludes failing machines, includes unrelated machines.



Future Work

- Test on additional data:
  - **Mobile network data.**
  - Key-value stores.
  - Hadoop logs.
- Sparse PCA.
- Infer jobs with subspace clustering.
- **Communication-efficient version.**
  - Distributed variance monitor to normalize data.
  - New class of PCA sketches.
  - Geometric monitoring based on convex decompositions.

- E. Liberty. Simple and deterministic matrix sketching. In Proc. KDD, 2013.
- M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. CoRR, 2014.
- M. Ghashami and J. M. Phillips. Relative errors for deterministic low-rank matrix approximations. In Proc. SODA. 2014.
- A. Lazerson, I. Sharfman, D. Keren, A. Schuster, M. Garofalakis, and V. Samoladas. Monitoring distributed streams using convex decompositions. In Proc. VLDB, 2015.