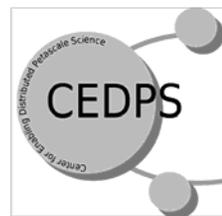




... for a brighter future

Anomaly Detection and Diagnosis in Grid Environments



*Lingyun Yang, Chuang Liu,
Jennifer M. Schopf, Ian
Foster*

*Argonne National Laboratory
University of Chicago*

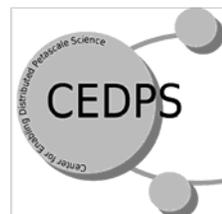


U.S. Department
of Energy

UChicago ►
Argonne_{LLC}

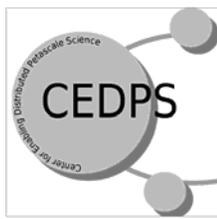
A U.S. Department of Energy laboratory
managed by UChicago Argonne, LLC

The Problem:



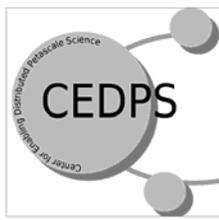
- Failure rates in production systems are extremely high
 - Both OSG and TG report 30% + failures at times
- Key to delivering reliable application-level is detection of anomalies
 - Unexpected changes in behavior
 - Reasons behind these changes
- Current techniques have a very high false positive rate
 - Some changes in performance are normal
 - *eg nightly backups*

Our Solution:



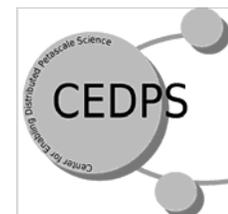
- 1) Filter out periodic data from trace data to identify real anomalies
- 2) A diagnostic technique to determine the cause of anomalies
- 3) Extensive experimentation
 - for CPU, memory-based, and network anomalies
 - simulated and actual environments

Anomaly Detection



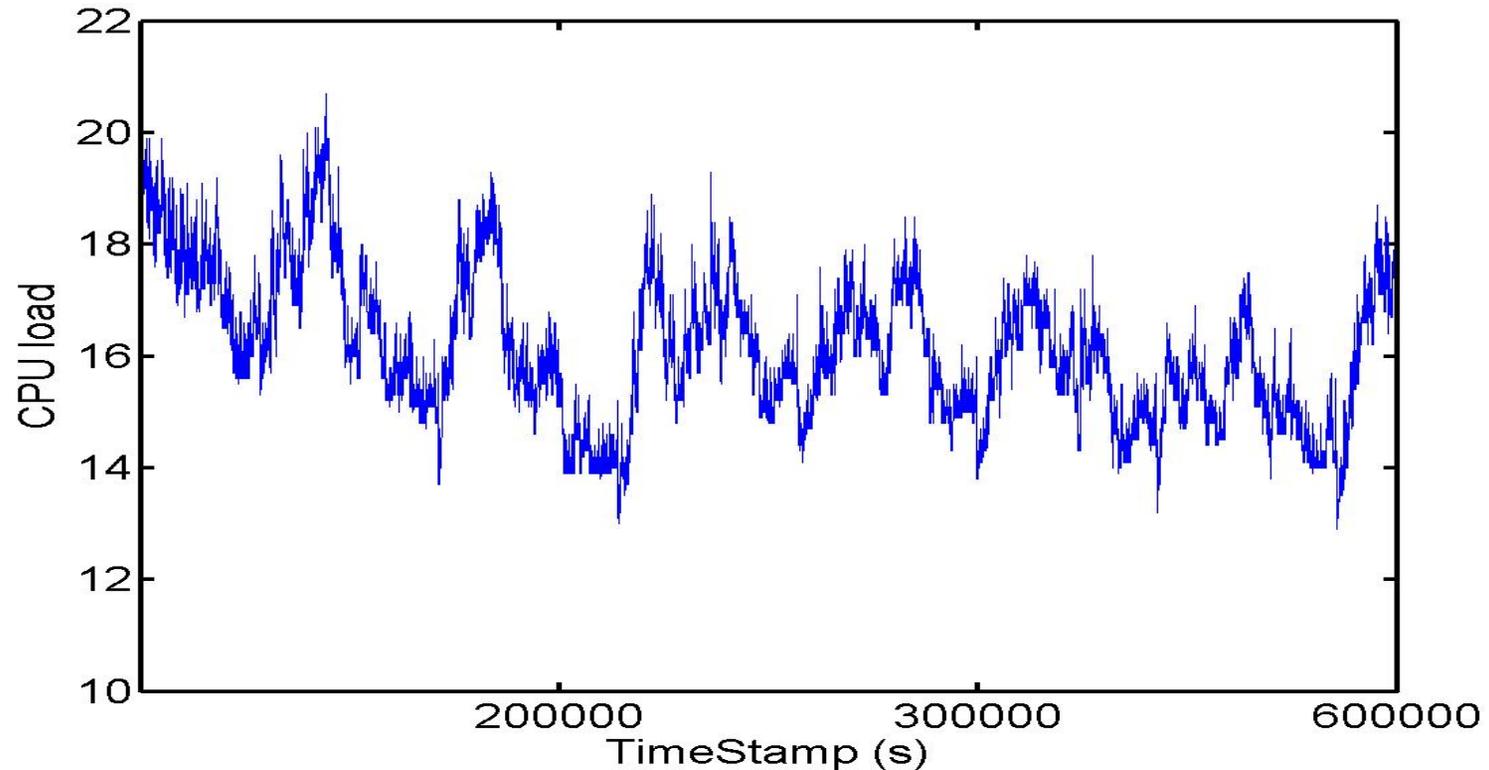
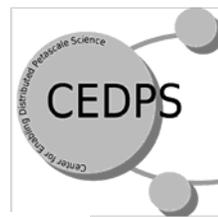
- True anomalies
 - Caused by a failure or unexpected event
- Periodic anomalies
 - Caused by normal periodic changes in behavior
 - For the most part should not be considered anomalous
- Only true anomalies should cause alert flags and be diagnosed

Periodic Behavior



- Shared network links have a daily pattern in that they are usually busier during the daytime
 - Barford, et al, 2002
- Number of jobs submissions has a 24-hour pattern of more daytime submissions and then a nightly draining of the queues
 - Downey, 1998
- Periodic system administration tasks can have finer-grained, even hourly, occurrences

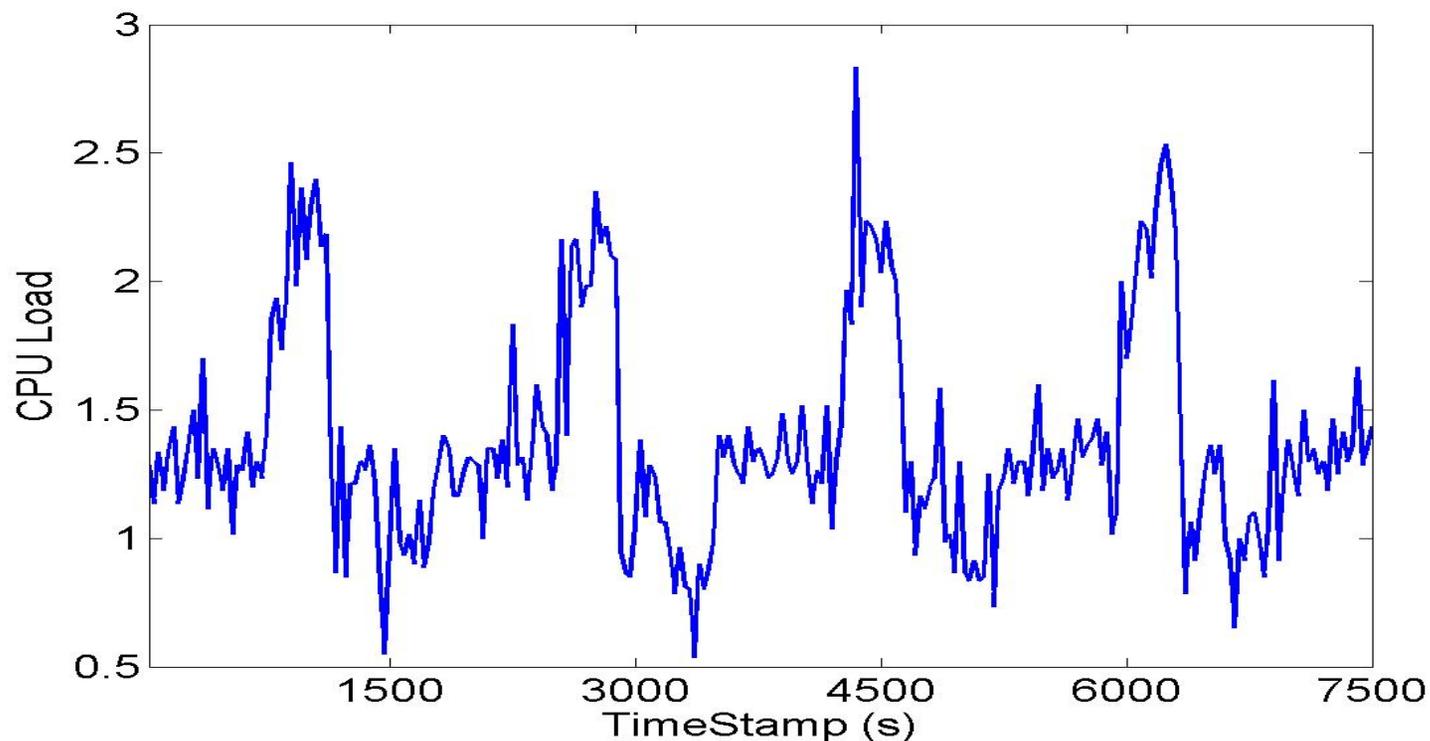
CPU Periodic Patterns



Week long trace showing daily periodic pattern of CPU load (righthand.eecs.harvard.edu)

Data measured every 30 seconds

CPU Periodic Patterns (2)



Two-hour CPU load trace showing half-hour variance
(vatos.cs.uchicago.edu)

Data measured every 30 seconds

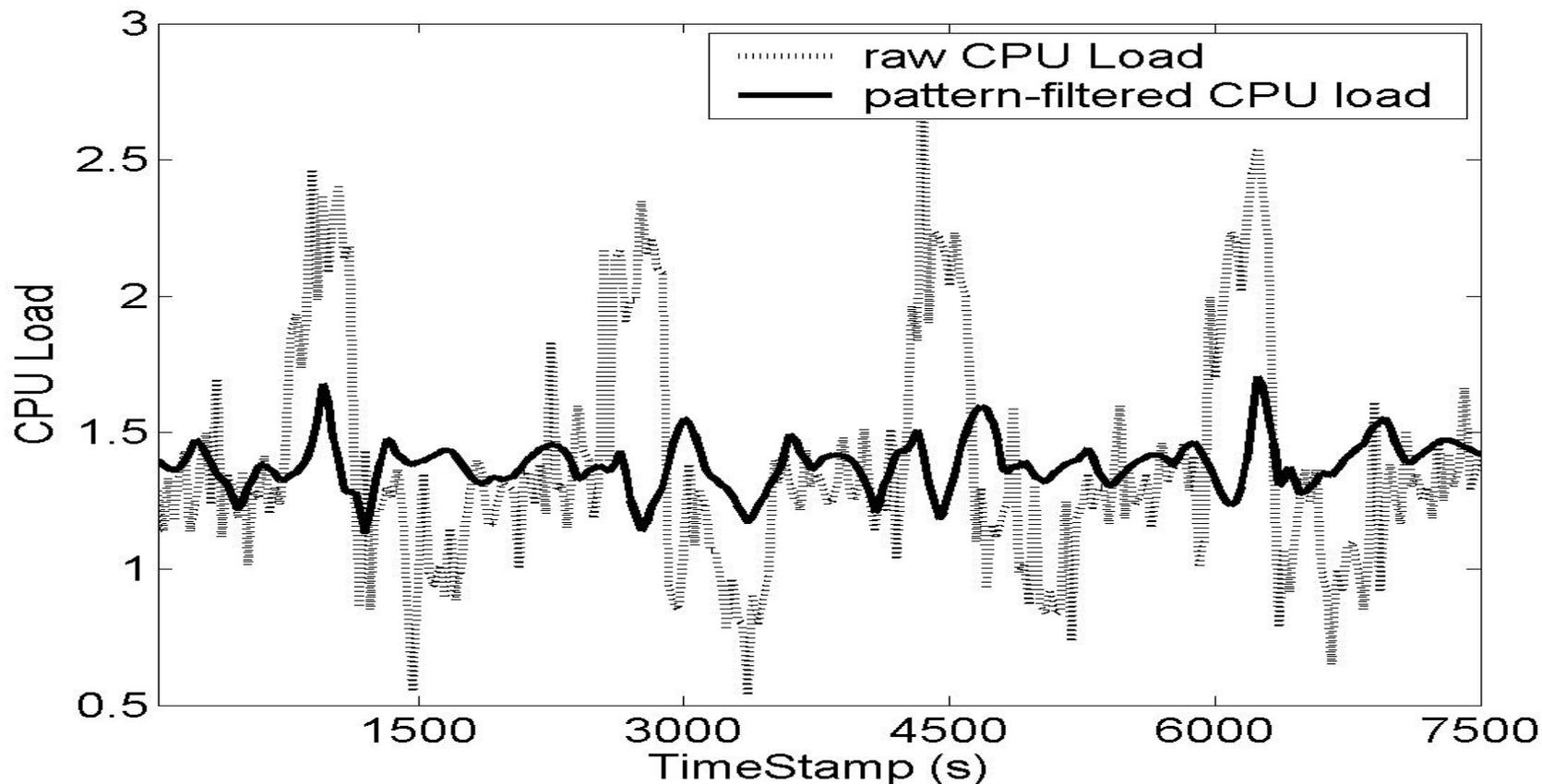
Traditional Anomaly Detection: Window Averaging

- Just what it sounds like
 - Average over a window
 - Anything greater or less than a threshold tagged as an anomaly
- Pros
 - Simple
 - Efficient
 - Widely used
- Cons
 - Can have a high false positive rate

Modified Window Averaging with Filtering

- Use Fourier transforms to filter the signal stream
 - Time domain representation shows how a signal changes over time
 - Frequency domain shows
 - *“Does the data include any periodic signals?”*
 - *“What is the frequency and amplitude if there are any?”*
- Remove all strong periodic signals
 - Set amplitude of the corresponding periodic frequency to zero
 - Transform back to the time domain
- **We do not need to know whether a periodic signal exists or what its frequency, amplitude, or shape is**

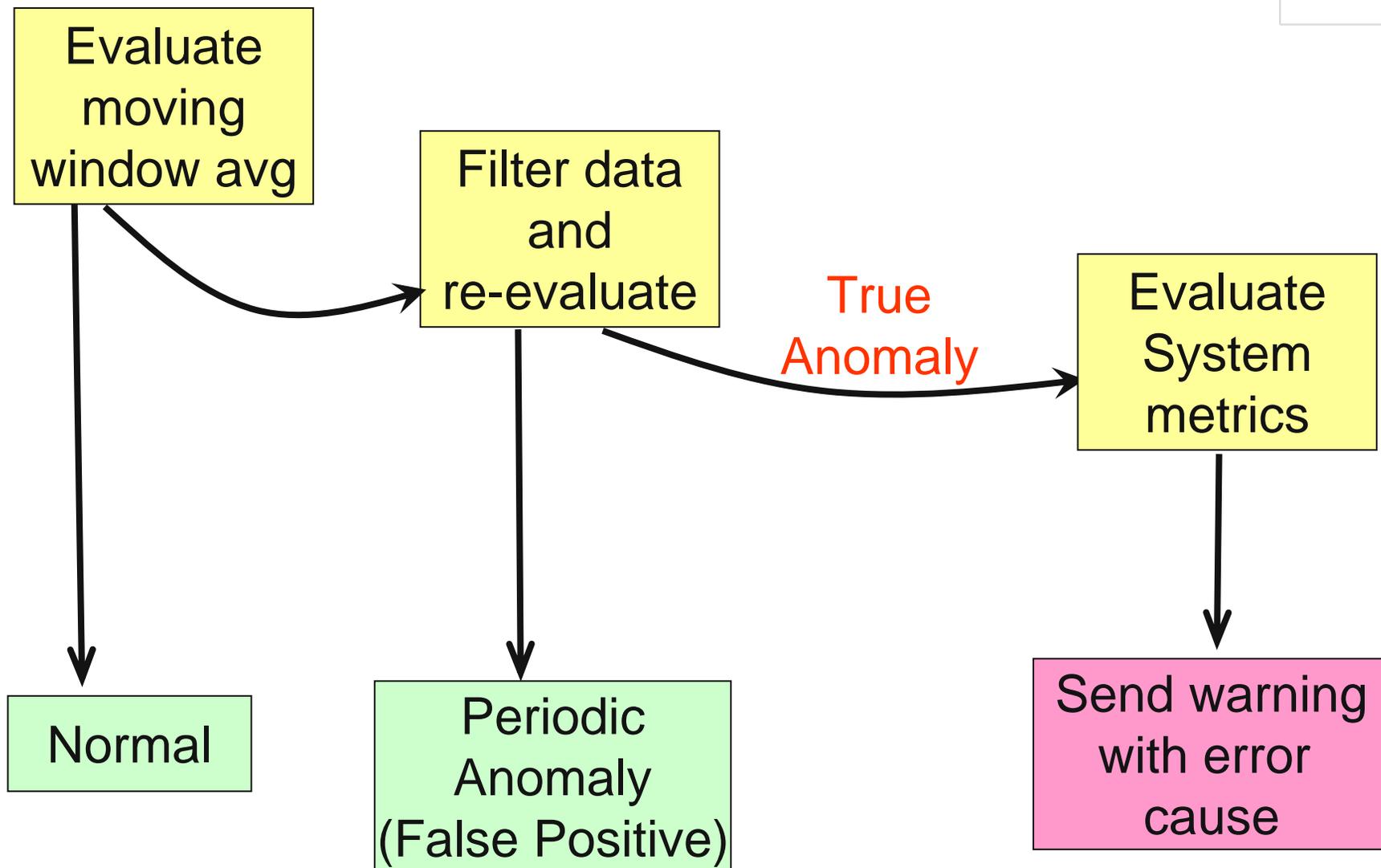
Filtering Example



Raw CPU load and CPU load after pattern-filtering

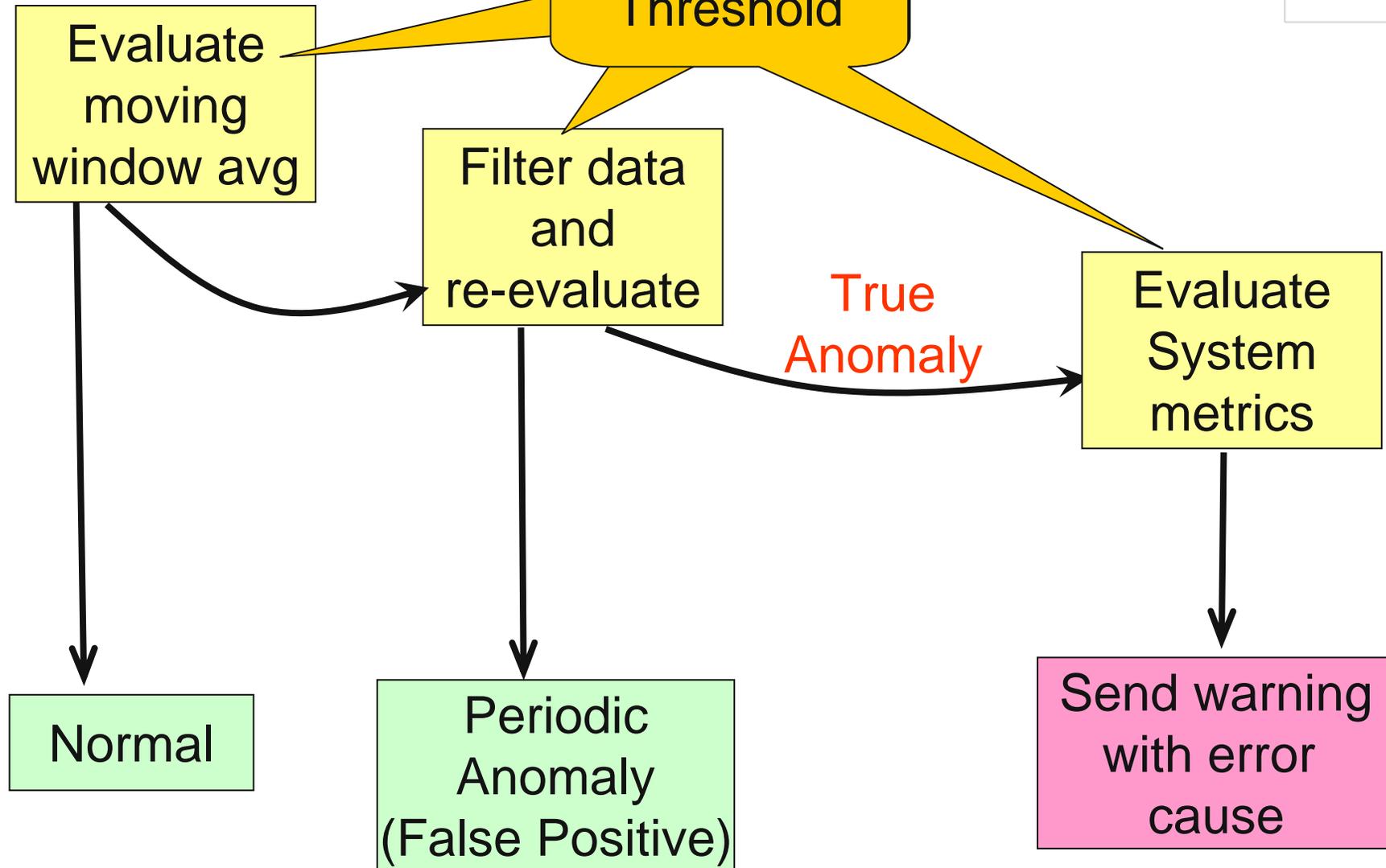
Anomaly Diagnostics

- In addition to application performance, collect system metrics for background behavior
 - Disk measurements
 - Network Weather Service BW measurements
 - Etc.
- Calculate the window average for each system metric
- Use this window average as the baseline for resource behavior
- Compare variance of system metrics with time of anomaly occurrence



Parameters

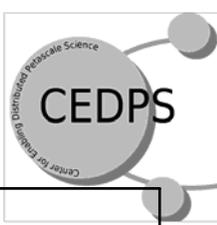
Window size
Threshold



Cactus Experiments

- Cactus on 4 node cluster
 - Evaluated changed in run time
- 4 sets of 2 week data
 - First set used as training data for parameters
- Inserted 100 anomalies
 - Bandwidth hog, memory hog and CPU hog programs
 - Random times
- Metrics:
 - Number of anomalies successfully detected
 - Number of false positives

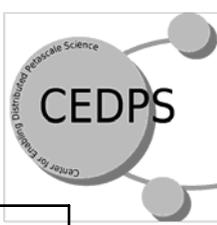
Cactus Detection Results



Data Set	Traditional		Modified	
	# of Hits	# of FPs	# of Hits	# of FPs
Data 1	99	588	96	63
Data 2	99	633	93	59
Data 3	98	551	94	89

- Traditional window average method
 - ~ 600 false positives
 - ~90% are caused by the half-hour periodic variations
- Modified approach
 - Eliminated between 84% and 91% of the false pos.
 - Identified between 93% and 96% of the injected anomalies

Cactus Diagnosis Results

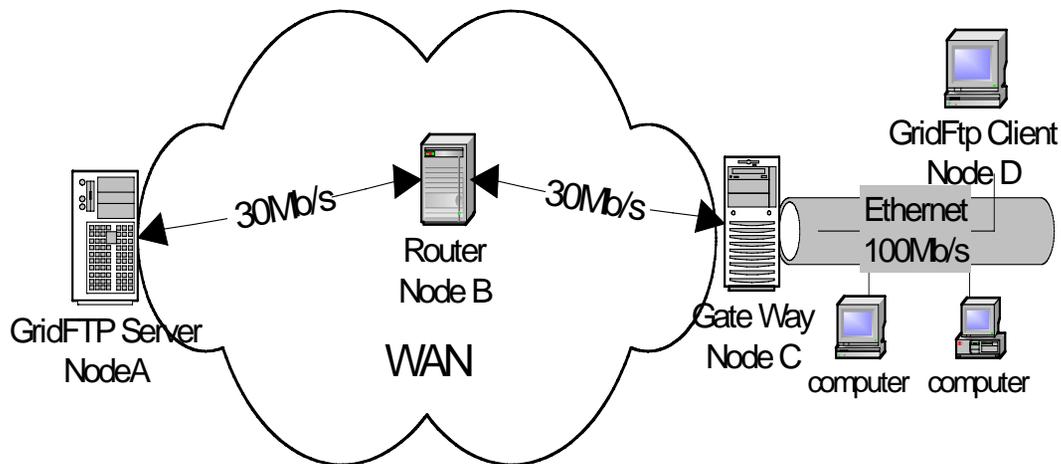


Data Set	# of Anomalies Detected	# of Anomalies Diagnosed
Data 1	96	87
Data 2	93	84
Data 3	94	82

- Compared the reasons reported with the type of anomaly inserted
 - CPU, Memory, Bandwidth
- Correct diagnosis in 82 to 87 of anomalies

GridFTP Data Transfers

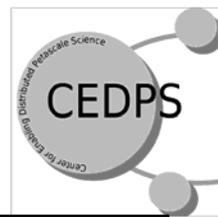
- Used Emulab so we could inject anomalies
- Collected resource performance data
 - GridFTP server and client machines
 - Ping measurements from the client and server node to other three nodes
- Performance metric for the GridFTP transfer is the data transfer rate, in megabits per second.



GridFTP Experiments (2)

- Three sets of GridFTP data
 - Each ~2 weeks
- 100 inserted anomalies across the three links in the path
 - Changed the traffic shaping parameters of each link in a random order
 - Decrease the bandwidth to less than 10% of orig
 - Increase delay (or loss ratio) by 5 to 10 times
 - CPU and memory anomalies did not effect perf
- Note: No periodic behaviors introduced

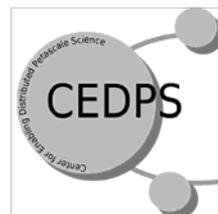
GridFTP Detection Results



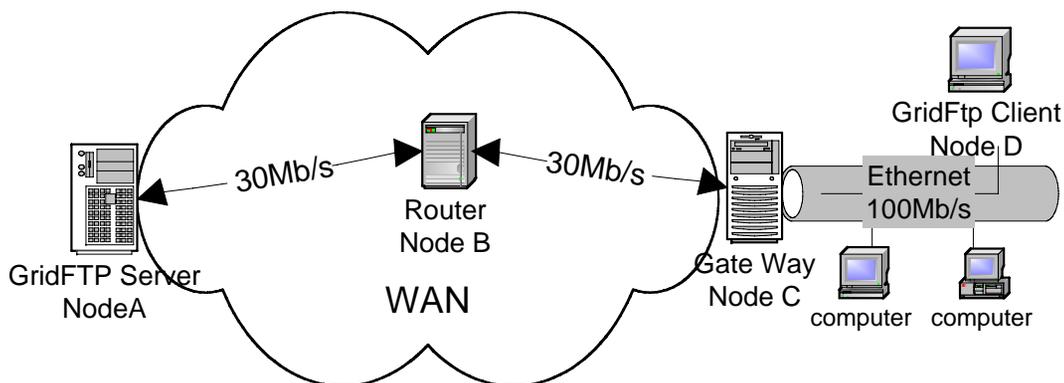
Data Set	Traditional		Modified	
	# of Hits	# of FPs	# of Hits	# of FPs
Data 1	99	5	92	2
Data 2	97	9	95	7
Data 3	100	6	90	4

- Our method is not as efficient as the traditional method for detecting anomalies when there is no periodic usage pattern in the resource performance

GridFTP Detection Results

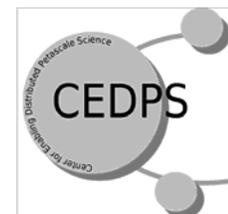


Data Set	# of Anomalies Detected	# of Anomalies Diagnosed
Data 1	92	73
Data 2	95	81
Data 3	90	74



For the 92 to 95 anomalies detected, our strategy finds the problematic links for 73 to 81 anomalies correctly

Sweep 3D



■ Sweep3D

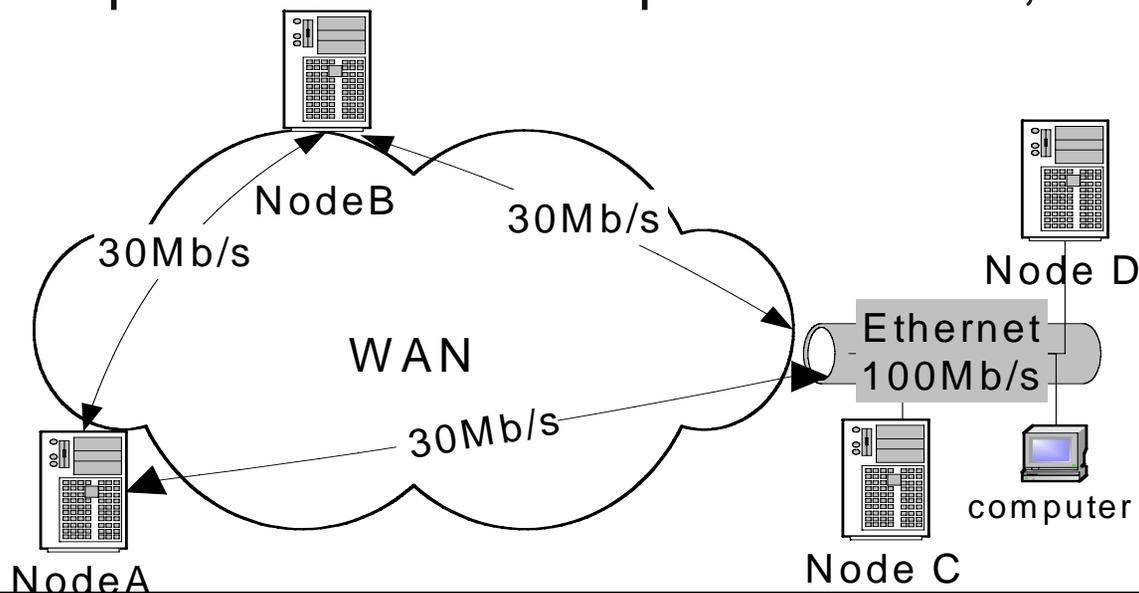
- 3D discrete ordinates neutron transport application
- Multiple processors using domain decomposition
- MPI message passing
- Execution includes both network communications and computation

■ With varying problem size, computation/communication ratio will change

- Application shifts from BW-bound to CPU-bound

Sweep3d Methodology

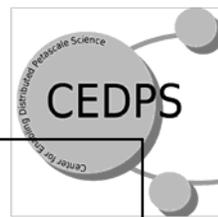
- Ran Sweep3d on Emulab to control the introduction of resource periodic patterns and anomalies
 - Four machines from three domains
 - One-dimensional decomposition
 - Communication happens on A-B, B-C, and C-D
- Emulated periodic CPU load patterns for A,B



Extra Simulated Behaviors

- Emulated periodic CPU load patterns for machines A,B
 - A had daily periodic CPU load pattern, amplitude equal to 5
 - B had 2 hourly periodic CPU load pattern with amplitude equal to 3
- Ran 3 different problem sizes
 - Each run ~9 days
 - Inserted 33,33,and 34 NW anomalies
 - Memory anomalies had no effect

Sweep3D Detection Results



Problem Size	Data Set	Traditional		Modified	
		# of Hits	# of FPs	# of Hits	# of FPs
Small	Data 1	33	1	31	0
	Data 2	33	4	29	3
	Data3	32	3	32	3
Medium	Data1	33	9	32	5
	Data 2	33	8	31	4
	Data 3	33	10	31	7
Large	Data 1	32	43	30	6
	Data 2	32	54	29	9
	Data 3	33	52	32	19

- Small - no statistical difference (comm bound)
- Medium – We detect half of the false positives, but miss 5 of 99 anom
- Large - Trad has 50 false positives ~90% caused by periodic behavior

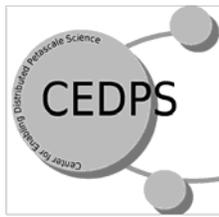
Sweep3D Diagnostic Results

Problem Size	Data Set	# of Anomalies Detected	# of Anomalies Diagnosed
Small	Data 1	31	30
	Data 2	29	28
	Data3	32	31
Medium	Data 1	32	31
	Data 2	31	28
	Data 3	31	27
Large	Data 1	30	28
	Data 2	29	29
	Data 3	32	31

Conclusions

- Periodic variations in resource performance is normal and inevitable
 - Can cause a high false positive rate for anomaly detection
- Our approach extends traditional methods by using signal processing techniques
 - Filter out periodic resource variation
 - Diagnosis technique to determine probable cause
- Experimental results show
 - Detection of up to 96% of anomalies
 - Reduction of false positive rate up to 90%
 - Diagnostic up to 70%

More Information



- Jennifer M. Schopf
 - jms@mcs.anl.gov
 - <http://www.mcs.anl.gov/~jms>
- Lingyun Yang
 - Now at Yahoo!
 - lyyang@yahoo-inc.com