



Towards Informatic Analysis of Syslogs

IEEE Cluster 2004
Sept 22

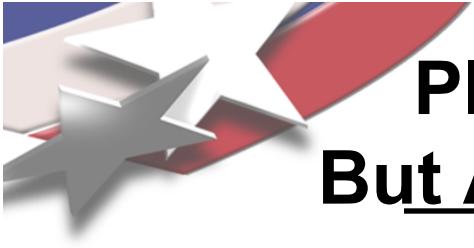
Jon Stearley
jrstear@sandia.gov

Sandia National Laboratories
Scalable Systems Integration Department



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.





Plenty of Faults, Plenty of Logs, But Actionable Information is Elusive

- Event logs are a ubiquitous source of system feedback, but are notoriously free in format
 - Supercomputers have many points of failure, with complex and dynamic interdependencies
- ∴ Identifying the root cause of faults in supercomputers is difficult (and expensive)

Leverage recent advances in informatics!



Log Analysis Background

	Monitoring (eg: regexps)	Visualization, Auralization
Expert effort required to generate	HIGH (for each new cluster or major upgrade)	LOW to MEDIUM
Clarity of understanding of results	HIGH	LOW to HIGH (often MEDIOCRE)
Examples	Swatch, Logsurfer, SEC,...	Xscal, Mielog, Peep,...

Log analysis is a well-studied but open problem.

Inspection of system logs is fundamental to debugging – increased capability to efficiently extract actionable information **WILL** impact MTTR (mean time to repair) and **MAY** impact MTBF (mean time between failure).



Charter

With the specific goal of increasing supercomputer RAS:

We intend to produce a machine-learning analysis system which enables content-novice analysts to

- efficiently understand evolving trends,**
- identify anomalies,**
- and investigate cause-effect hypotheses in large multiple-source log sets.**

RAS = Reliability, Availability, and Serviceability



Analyst Thought Process

This presentation

Ongoing research

“What aspects of the message stream are strongly correlated with system malfunction or misuse?”

1. What message content **and occurrence rate** is normal?
2. What message groups are normal?
3. What is new?
4. **What is missing?**

5. Can devices be classified by their output message stream?
6. Can users or applications be classified by their resulting message stream?
7. Are device-to-device and/or job-to-job log stream similarities sufficient to identify hardware or software failures?



Sisyphus Toolkit Approach

2. Group time-correlated message types into message blocks
3. Analyze message types and blocks over time, by source, job, etc

1. Convert word sequences into message types

```
S84 ← PCT-553[430]: app process done, exit code 1, terminating signal 0
S0 ← TSUNAMI machine check: vector=0x630 pc=0xfffffc000032f310 code=0x100000086
S0 ← machine check type: correctable ECC error (retryable)
{ S14 in.rshd[928]: connect from 192.168.254.3
  S0 pam_rhosts_auth[928]: allowed to root@admin-0 as root
  S44 PAM_pwdb[928]: (rsh) session opened for user root by (uid=0)
  S0 in.rshd[929]: root@admin-0 as root: cmd='/bin>true'
  S84 PCT-537[430]: app process done, exit code 1, terminating signal 0
  S84 PCT-541[430]: app process done, exit code 1, terminating signal 0
  S6 PCT-540[430]: ignoring ABORT_LOAD FIRST TRY from 797/2, unknown job ID 1733
  S5 nfs: task 64052 can't get a request slot
  { S14 in.rshd[948]: connect from 192.168.254.3
    S0 pam_rhosts_auth[948]: allowed to root@admin-0 as root
    S44 PAM_pwdb[948]: (rsh) session opened for user root by (uid=0)
    S0 in.rshd[949]: root@admin-0 as root: cmd='/cluster/bin/power --cycle node.n-18.t-37 node.n-15.t-37 node.n-27.t-37 node.n-12.t-37 node.n-24.t-37 node.n-21.t-37 node.n-30.t-37 node.n-19.t-37 node.n-4.t-37 node.n-16.t-37 node.n-28.t-37 node.n-1.t-37 node.n-13.t-37 node.n-25 10.t-37 node.n-22.t-37 node.n-31.t-37 node.n-8.t-37 node.n-5.t-37 node.n-17.t-37 node.n-29.t-37 no node.n-14.t-37 node.n-26.t-37 node.n-11.t-37 node.n-23.t-37 node.n-20.t-37 node.n-32.t-37 node.n-node.n-6.t-37'
    S77 /sbin/mingetty[433]: console: invalid character □ in login name
    S77 /sbin/mingetty[433]: console: invalid character □ in login name
    S77 /sbin/mingetty[433]: console: invalid character □ in login name
    S21 tulip.c:v0.91g-ppc 7/16/99 becker@cesdis.gsfc.nasa.gov
    S87 eth0: Digital DS21143 Tulip rev 65 at 0x8000, 08:00:2B:87:1A:16, IRQ 29.
    S53 eth0: EEPROM default media type Autosense.
    S100 eth0: Index #0 - Media 10baseT (#0) described by a 21142 Serial PHY (2) block.
```



Automated Message Typing: Learning from Teiresias

What message content is normal?

Teiresias (Downloadable bioinformatics code from IBM TJ Watson), has two stages of operation:

1. **Scanning:** enumerate all elementary patterns of at least L/W specificity (i.e. find all phrases of length W words, where at least L of them never change (the others do change))
2. **Convolution:** combine elementary patterns into maximal irredundant “motifs” (leverage property of “downward closure”). Output motifs in decreasing specificity and support order.

Action: use teiresias-int to generate message motifs (word-to-int conversions necessary), compute time statistics, and present in logview (`teirify`)

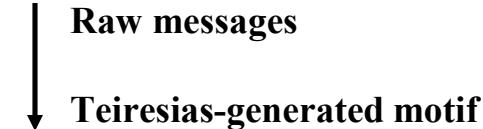
Result: automatically-generated “message phrases” sorted into three categories: common, deviant, anomalous

Note: medium probability of producing phrases which are common to multiple messages!



Example Message Type

```
Dec 19 06:41:18 dnsserver named[285]: XSTATS 1008769278 1007338854 RR=345808 RNXD=28957 RFwdR=210938 RDupR=736
RFail=933 RFErr=0 RErr=224 RAXFR=76 RLame=8441 ROpts=0 SSysQ=75649 SAns=2538210 SFwdQ=205371 SDupQ=48062
SErr=0 RQ=2576086 RIQ=5 RFwdQ=205371 RDupQ=4351 RTCP=18069 SFwdR=210938 SFail=11 SFErr=0 SNaAns=430699
SNXD=131184 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5939
Dec 19 07:41:18 dnsserver named[285]: XSTATS 1008772878 1007338854 RR=346283 RNXD=28981 RFwdR=211187 RDupR=737
RFail=933 RFErr=0 RErr=224 RAXFR=76 RLame=8460 ROpts=0 SSysQ=75802 SAns=2545771 SFwdQ=205588 SDupQ=48083
SErr=0 RQ=2583687 RIQ=5 RFwdQ=205588 RDupQ=4352 RTCP=18072 SFwdR=211187 SFail=11 SFErr=0 SNaAns=432300
SNXD=131615 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5940
Dec 19 08:41:18 dnsserver named[285]: XSTATS 1008776478 1007338854 RR=346834 RNXD=29027 RFwdR=211484 RDupR=737
RFail=933 RFErr=0 RErr=224 RAXFR=76 RLame=8485 ROpts=0 SSysQ=75976 SAns=2549937 SFwdQ=205864 SDupQ=48156
SErr=0 RQ=2587897 RIQ=5 RFwdQ=205864 RDupQ=4356 RTCP=18081 SFwdR=211484 SFail=11 SFErr=0 SNaAns=432827
SNXD=131790 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5943
Dec 19 09:41:18 dnsserver named[285]: XSTATS 1008780078 1007338854 RR=347741 RNXD=29093 RFwdR=212026 RDupR=741
RFail=938 RFErr=0 RErr=224 RAXFR=76 RLame=8517 ROpts=0 SSysQ=76179 SAns=2555207 SFwdQ=206373 SDupQ=48262
SErr=0 RQ=2593234 RIQ=5 RFwdQ=206373 RDupQ=4363 RTCP=18090 SFwdR=212026 SFail=11 SFErr=0 SNaAns=433461
SNXD=132082 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5946
Dec 19 10:41:18 dnsserver named[285]: XSTATS 1008783678 1007338854 RR=348920 RNXD=29180 RFwdR=212702 RDupR=745
RFail=947 RFErr=0 RErr=224 RAXFR=76 RLame=8588 ROpts=0 SSysQ=76470 SAns=2560964 SFwdQ=206994 SDupQ=48432
SErr=0 RQ=2599107 RIQ=5 RFwdQ=206994 RDupQ=4379 RTCP=18099 SFwdR=212702 SFail=12 SFErr=0 SNaAns=434176
SNXD=132398 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5949
Dec 19 11:41:18 dnsserver named[285]: XSTATS 1008787278 1007338854 RR=350302 RNXD=29233 RFwdR=213504 RDupR=746
RFail=948 RFErr=0 RErr=224 RAXFR=76 RLame=8648 ROpts=0 SSysQ=76767 SAns=2567426 SFwdQ=207803 SDupQ=48567
SErr=0 RQ=2605665 RIQ=5 RFwdQ=207803 RDupQ=4394 RTCP=18105 SFwdR=213504 SFail=12 SFErr=0 SNaAns=435334
SNXD=132774 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5951
Dec 19 12:41:18 dnsserver named[285]: XSTATS 1008790878 1007338854 RR=351491 RNXD=29297 RFwdR=214187 RDupR=750
RFail=948 RFErr=0 RErr=226 RAXFR=76 RLame=8701 ROpts=0 SSysQ=77062 SAns=2574008 SFwdQ=208484 SDupQ=48720
SErr=0 RQ=2612350 RIQ=5 RFwdQ=208484 RDupQ=4409 RTCP=18108 SFwdR=214187 SFail=13 SFErr=0 SNaAns=436277
SNXD=133143 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5952
```



```
named: XSTATS * 1007338854 RR=* RNXD=* RFwdR=* RDupR=* RFail=* RFErr=0 RErr=* RAXFR=76 RLame=*
ROpts=0 SSysQ=* SAns=* SFwdQ=* SDupQ=* SErr=0 RQ=* RIQ=5 RFwdQ=* RDupQ=* RTCP=* SFwdR=* SFail=*
SFErr=0 SNaAns=* SNXD=* RUQ=0 RURQ=0 RUXFR=0 RUUpd=
```



Message Type Categorization

Teiresias' output sort by decreasing specificity and support enables message type categorization of increasing anomaly:

1. **Common** – message types which occur at least k times (frequent)
2. **Deviant** – messages which appear fewer than k times (infrequent), but are similar in content to frequent messages
3. **Anomalous** – messages which are completely anomalous in content and occurrence



Automated Message Typing: Leveraging SLCT

Simple Logfile Clustering Tool (SLCT), operates in three phases:

1. **Frequent words** – hash and count all {position,word} tuples (pw), prune those occurring less than k times (**most words in logs are not frequent!**)
2. **Frequent word clusters** – has and count all single-line pw “clusters” (“1w 2w 3_* 4_w”, ie message types), prune those occurring less than k times
3. **Outliers and wildcard refinement (optional)** – determine “outlier” messages (anomalies), and determine constant prefix or suffix for wildcards

Output categorization (and review is very tedious):

- word clusters occurring at least k times (**common**)
- less specific pw clusters which, if joined with above pw clusters, occur at least k times (**common+deviant**)
- lines not matching either of the above (**anomalous**)

Note: use of word position results in lower probability of producing phrases common to multiple messages!



Example Message Types

```
Dec 19 06:41:18 dnsserver named[285]: XSTATS 1008769278 1007338854 RR=345808 RNXD=28957 RFwdR=210938 RDupR=736 RFail=933  
RFErr=0 RErr=224 RAXFR=76 RLame=8441 ROpts=0 SSysQ=75649 SAns=2538210 SFwdQ=205371 SDupQ=48062 Serr=0 RQ=2576086 RIQ=5  
RFwdQ=205371 RDupQ=4351 RTCP=18069 SFwdR=210938 SFail=11 SFErr=0 SNaAns=430699 SNXD=131184 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5939  
Dec 19 07:41:18 dnsserver named[285]: XSTATS 1008772878 1007338854 RR=346283 RNXD=28981 RFwdR=211187 RDupR=737 RFail=933  
RFErr=0 RErr=224 RAXFR=76 RLame=8460 ROpts=0 SSysQ=75802 SAns=2545771 SFwdQ=205588 SDupQ=48083 Serr=0 RQ=2583687 RIQ=5  
RFwdQ=205588 RDupQ=4352 RTCP=18072 SFwdR=211187 SFail=11 SFErr=0 SNaAns=432300 SNXD=131615 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5940  
Dec 19 08:41:18 dnsserver named[285]: XSTATS 1008776478 1007338854 RR=346834 RNXD=29027 RFwdR=211484 RDupR=737 RFail=933  
RFErr=0 RErr=224 RAXFR=76 RLame=8485 ROpts=0 SSysQ=75976 SAns=2549937 SFwdQ=205864 SDupQ=48156 Serr=0 RQ=2587897 RIQ=5  
RFwdQ=205864 RDupQ=4356 RTCP=18081 SFwdR=211484 SFail=11 SFErr=0 SNaAns=432827 SNXD=131790 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5943  
Dec 19 09:41:18 dnsserver named[285]: XSTATS 1008780078 1007338854 RR=347741 RNXD=29093 RFwdR=212026 RDupR=741 RFail=938  
RFErr=0 RErr=224 RAXFR=76 RLame=8517 ROpts=0 SSysQ=76179 SAns=2555207 SFwdQ=206373 SDupQ=48262 Serr=0 RQ=2593234 RIQ=5  
RFwdQ=206373 RDupQ=4363 RTCP=18090 SFwdR=212026 SFail=11 SFErr=0 SNaAns=433461 SNXD=132082 RUQ=0 RURQ=0 RUXFR=0 RUUpd=5946
```

“frequent-itemset mining”

`teirify`:

```
named: XSTATS * 1007338854 RR=* RNXD=* RFwdR=* RDupR=* RFail=* RFErr=0  
RErr=* RAXFR=76 RLame=* ROpts=0 SSysQ=* SAns=* SFwdQ=* SDupQ=* SErr=0  
RQ=* RIQ=5 RFwdQ=* RDupQ=* RTCP=* SFwdR=* SFail=* SFErr=0 SNaAns=*  
SNXD=* RUQ=0 RURQ=0 RUXFR=0 RUUpd=
```

`slct`:

```
named[285]: XSTATS 1008*78 1007338854 RR=3* RNXD=2* RFwdR=21* RDupR=7* * RFErr=0  
* RAXFR=76 RLame=8* ROpts=0 SSysQ=7* SAns=2* SFwdQ=2* SDupQ=4* SErr=0 RQ=2*  
RIQ=5 RFwdQ=2* RDupQ=4* RTCP=18* SFwdR=21* * SFErr=0 SNaAns=4* SNXD=13* RUQ=0  
RURQ=0 RUXFR=0 RUUpd=59*
```



teirify vs slct

	teirify	slct
Computational Complexity	Combinatorial in number of unique word sequences. Memory bottlenecked!!!	Linear in number of total words. Memory-efficient and MUCH faster!
Message typing effectiveness	Good (all motifs meet required specificity, but multi-line phrases are confusing)	Great (no multi-line phrases, but some message types are uselessly vague)
Source Code	Closed	Open
Number of input parameters	teirify L W k logfile	slct -s k -b B -f '[^[:space:]] (+)' -t '\$1' -r -j logfile
Message typing categorization	Excellent (common, deviant, anomalous)	Fair (common, common+deviant, anomalous)
Output sort	Good (by decreasing support and specificity)	Poor (increasing specificity only)
Ease of reviewing results	Good (logview)	Poor

slct: effective, but difficult to use and difficult to review results



teirify vs slctit

	teirify	slctit
Computational Complexity	Combinatorial in number of unique word sequences. Memory bottlenecked!!!	Linear in number of total words. Memory-efficient and MUCH faster!
Message typing effectiveness	Good (all motifs meet required specificity, but multi-line phrases are confusing)	Great (no multi-line phrases, but some message types are uselessly vague)
Source Code	Closed	Open
Number of input parameters	teirify L W k logfile	slctit k logfile
Message typing categorization	Excellent (common, deviant, anomalous)	Excellent (common, deviant, anomalous)
Output sort	Good (by decreasing support and specificity)	Great (decreasing support and specificity, by time)
Ease of reviewing results	Good (logview)	Good (logview)

#	label	k	period	stddev	motif
L115		32	1	3	HWRPB cycle frequency (462962962) seems inaccurate - usin
L75		26	1	1	rte-init: Found a LANai type 7.2 with 2097152 bytes (20
L76		6	3	6	rte-init: Found a LANai type * with 2097152 bytes (2048
L57		13	3600	0	named: XSTATS * 1007338854 RR=* RNXD=* RFwdR=* RDupR=* R
LO		44	0	0	NOCLASS

(Logview snapshot)

View lines in original (ungrouped) order

```

L75 Nov 25 17:53:25 src@node/if-0.n-3.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:25 src@node/if-0.n-23.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:25 src@node/if-0.n-4.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:27 src@node/if-0.n-11.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:29 src@node/if-0.n-2.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:29 src@node/if-0.n-17.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:30 src@node/if-0.n-9.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:30 src@node/if-0.n-22.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:31 src@node/if-0.n-5.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with
L75 Nov 25 17:53:31 src@node/if-0.n-8.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 7.2 with

L76
L76 Nov 25 17:53:04 src@node/if-0.n-15.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with
L76 Nov 25 17:53:07 src@node/if-0.n-20.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with
L76 Nov 25 17:53:23 src@node/if-0.n-28.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with
L76 Nov 25 17:53:24 src@node/if-0.n-32.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with
L76 Nov 25 17:53:25 src@node/if-0.n-25.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with
L76 Nov 25 17:53:31 src@node/if-0.n-29.t-37/if-1.n-0.t-37 rte-init: Found a LANai type 9.0 with

LO
LO Nov 25 00:22:26 src@node/if-0.n-28.t-37/if-1.n-0.t-37 TSUNAMI machine check: vector=0x630 pc
LO Nov 25 00:22:26 src@node/if-0.n-28.t-37/if-1.n-0.t-37 machine check type: correctable ECC er
LO Nov 25 06:06:15 src@node/if-0.n-5.t-37/if-1.n-0.t-37 PCT-540[430]: ignoring ABORT_LOAD FIRST
LO Nov 25 06:09:18 src@node/if-0.n-20.t-37/if-1.n-0.t-37 nfs: task 64052 can't get a request sl

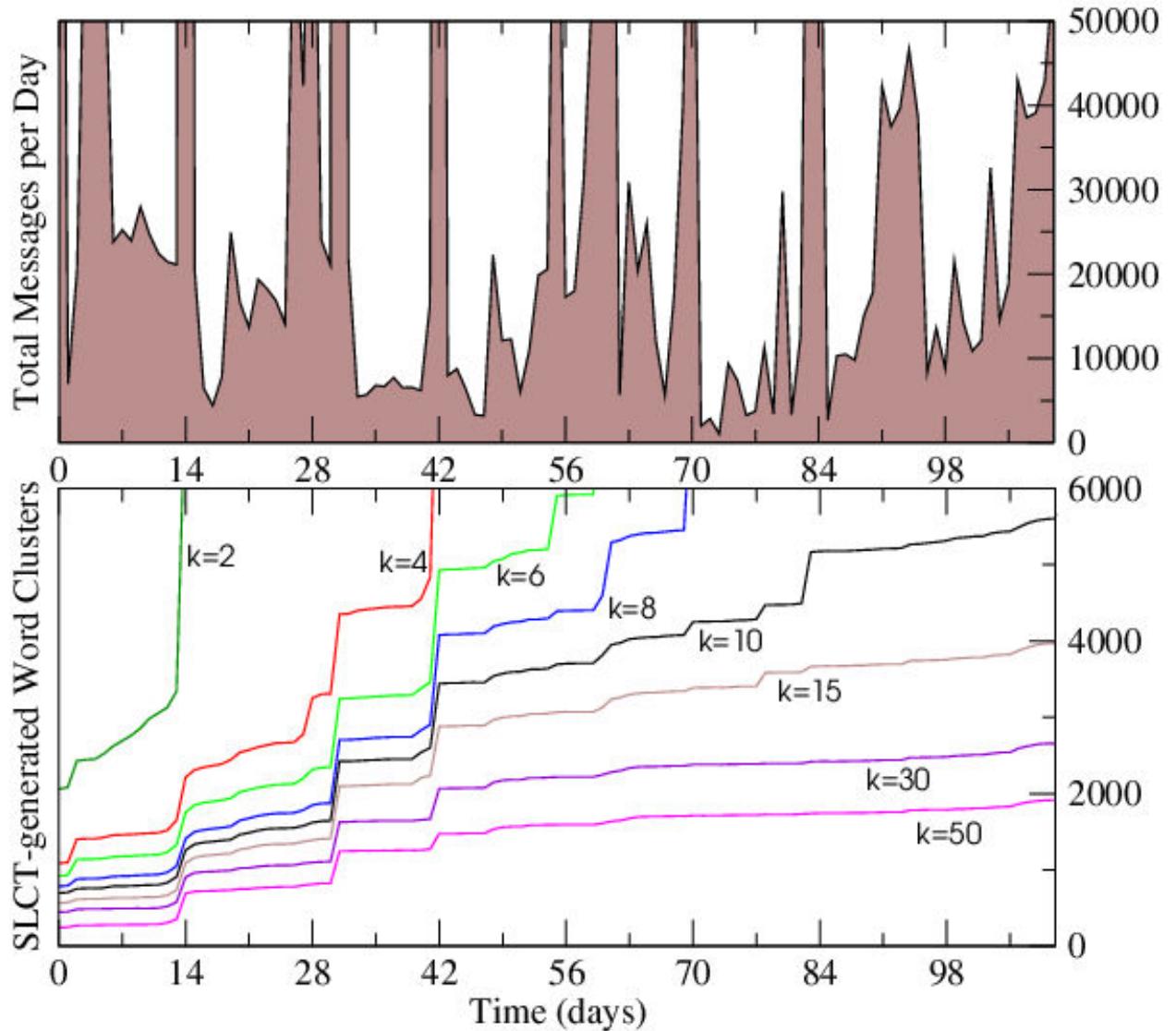
```



Message Type Novelty Rate

Not all periods of high raw message count result in new message types.

**Message type novelty rate is the number of new message types per unit time (dk/dt), and provides an answer to the question:
“anything new?”**





Conclusion

This work provides incremental contributions towards

- the efficient review of event logs**
- the efficient establishment of optimal monitoring rules**
- a basis for performing anomaly detection in event logs**



Future Work

Active

- Further exploration of novelty rate threshold (dk/dt)
- Analysis of SLCT-generated message types over time, source, job, etc to “investigate cause-effect hypotheses” ie:
 - “what is the most anomalous hour last week (and why)?”
 - “what is the most anomalous host in the cluster (and why)?”
 - “what (if any) message types are particularly correlated with the list of (fault) times (t_1, t_2, t_3) or application runs (a_1, a_2, a_3)?”

Proposed

- Present Deviant message types adjacent to their Common “parent” (in decreasing specificity order)
- Apply minimum-specificity threshold
- Automatic import of SLCT-generated message types into log monitoring software (ie logsurfer or SEC)
- Automatic determination of support threshold or novelty rate threshold based on hypothesis being evaluated



Thanks and Pointer

Many thanks to:

**Nathan Dauchy, George Davidson, Tim Draelos,
Kevin Boyack, Risto Vaarandi, Donna Johnson,
Jerry Smith**

**Sisyphus toolkit is pending LGPL-approval. Send
jrstear@sandia.gov email if interested, and/or watch
<http://www.cs.sandia.gov> “Open-Source Software
Downloads” link (in “Select software package” pull-
down).**



Backup Slides

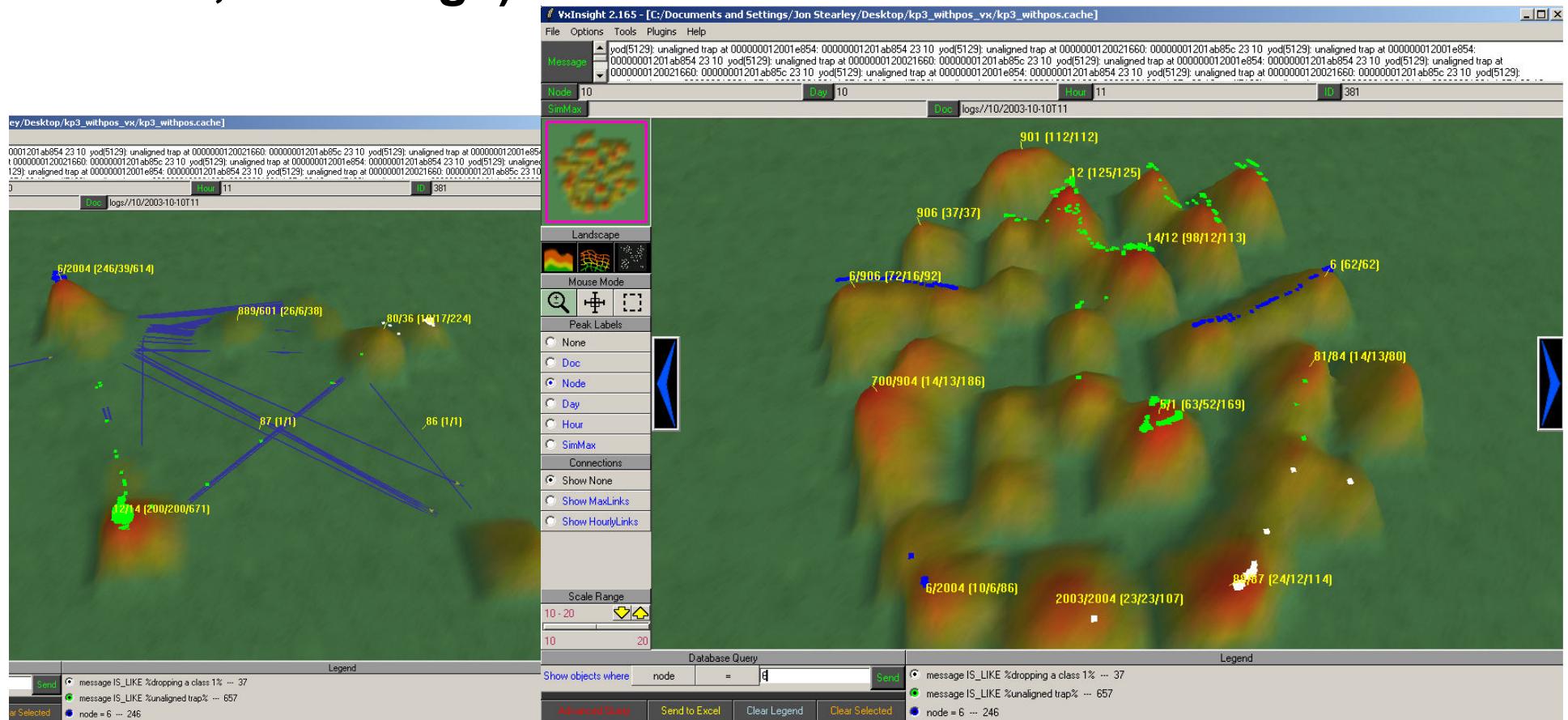
Problem Statement





Event-Logs-LSA Teaser

Below are snapshots of LSA results on Cplant syslogs, visualized with VxInsight, a data mining application used for patent and gene research (and now, event logs)

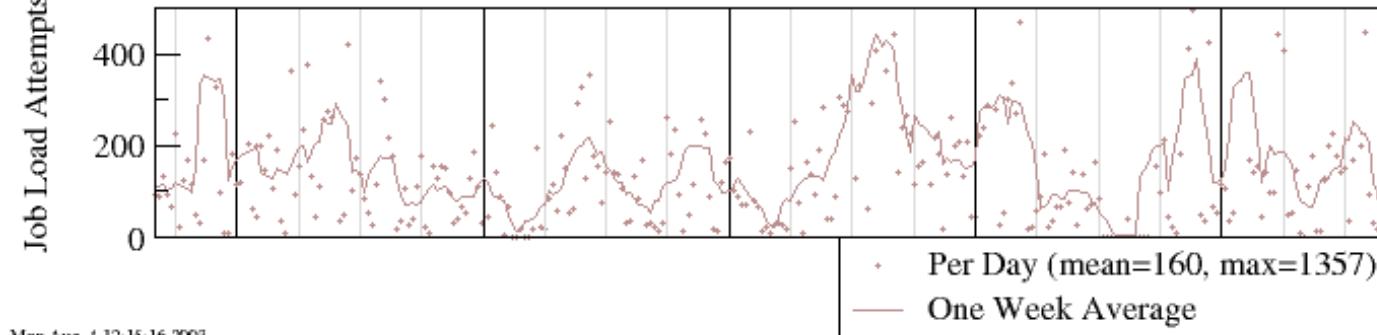
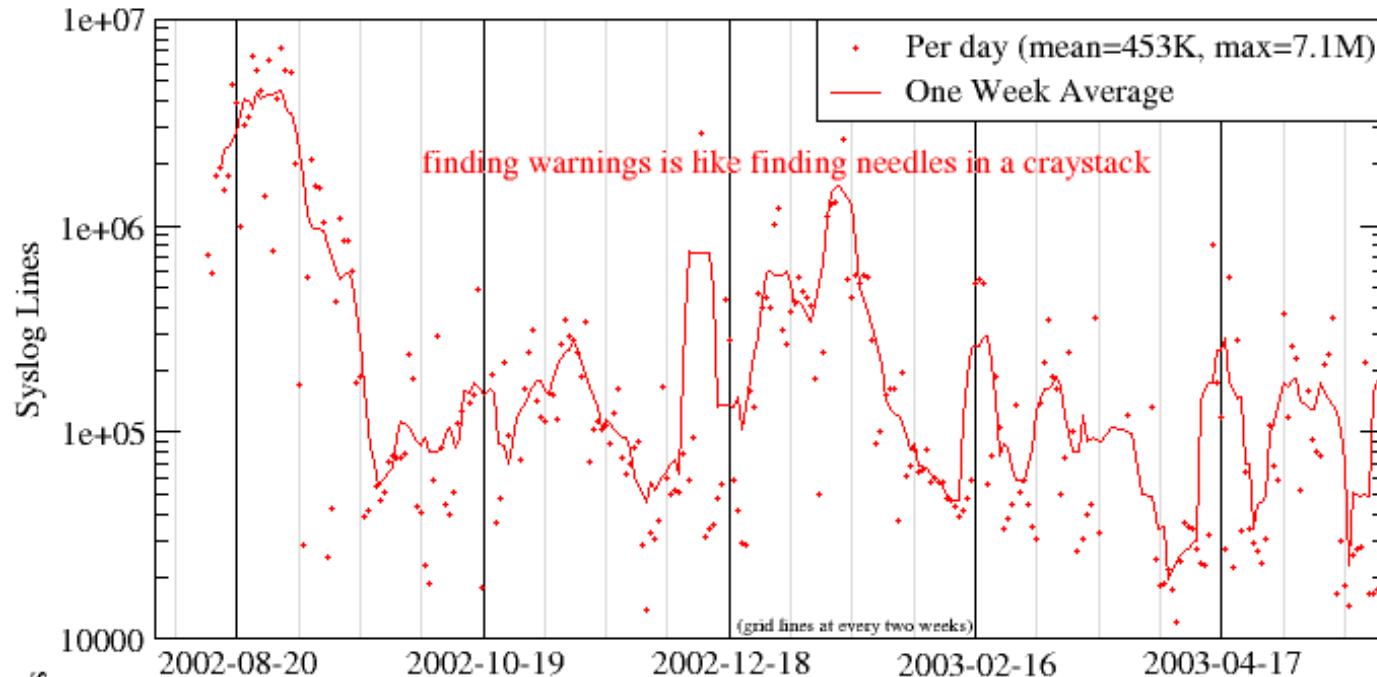




Lots of Logs

Ross Syslog Lines

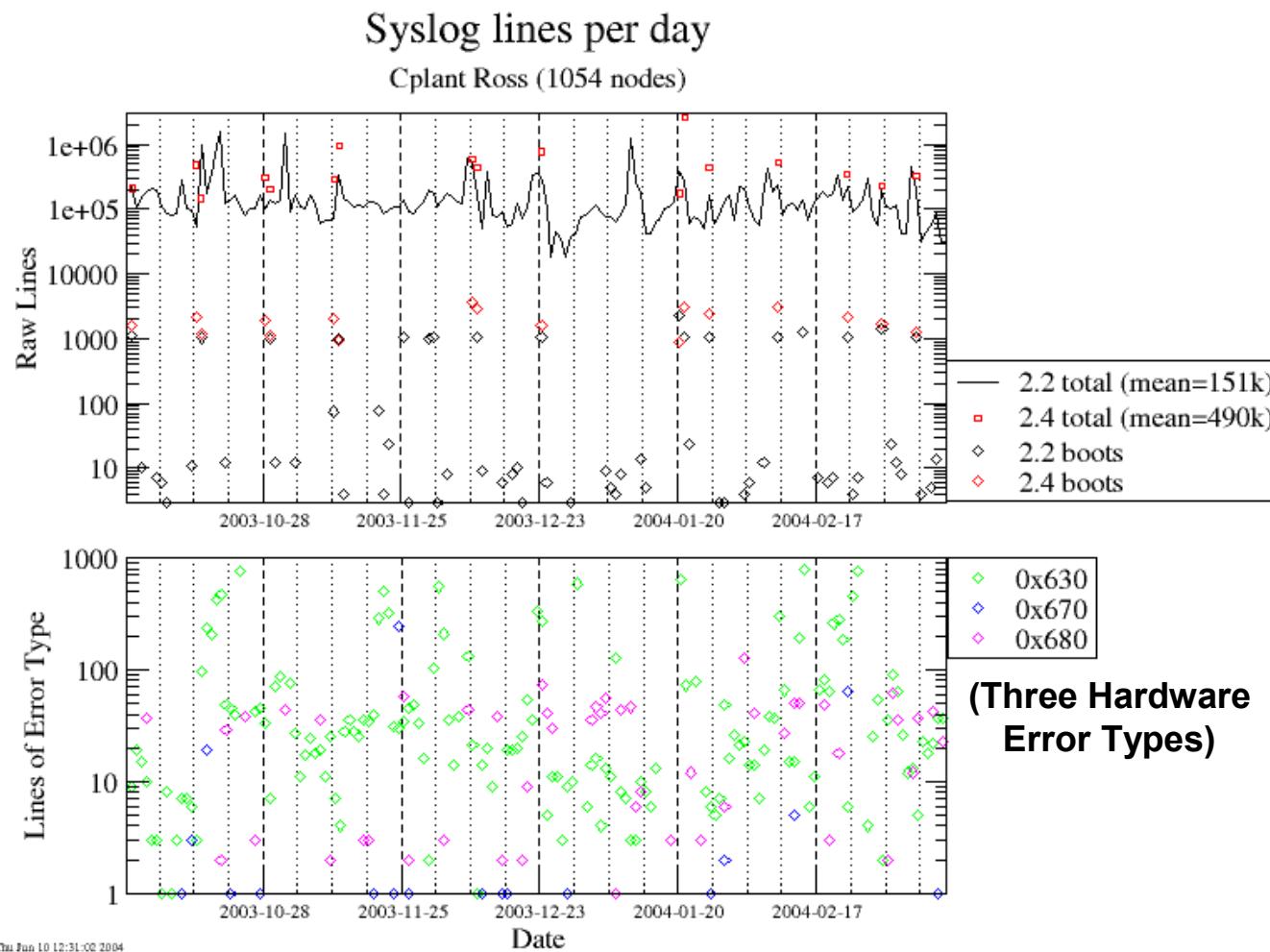
(roughly periodic with biweekly system reboots, slight correlation with job turnover rate)



Problem Statement



Have Established a Fault-Annotated Log Database (Three Years of Cplant)



SQL Database supports flexible subset selection:

- By Device
- By Time
- By Job
- By User
- By Error or Message Type
- etc



Automated Grouping of Time-correlated Messages

What message groups are normal?

Observation:

Rigid time-correlated message sequences will occur the same number of times, with the same time properties.

Approach:

For each message type:

- calculate support (k) and inter-arrival histogram
- present inter-arrival median and stddev
- (future work: cluster by histogram)

Result:

Exposes periodic message types and grouping into correlated message blocks



Automated Grouping of Time-correlated Messages

#	label	k	mean	stddev	motif
L28		33	1	180	ttyS00 at 0x03f8 (irq =4) is a 16550A
L33		33	1	180	Buffer cache hash table entries: 524288 (order 9, 4096k)
L38		33	1	180	TCP: Hash tables configured (ehash 524288 bhash 65536)
L48		33	1	180	eth0: Digital DS21143 Tulip rev 65 at 0x8000, * IRQ 29.
L53		33	1	180	eth1: Digital DS21143 Tulip rev 65 at 0x8800, * IRQ 30.
L124		33	1	180	host=* domain=, nis-domain=(none),
L127		33	1	180	Freeing unused kernel memory: 296k freed
L5		33	0	3	tftpd: tftpd: trying to get file: /tftpboot/vmlinuz-alpha.cluster
L95		32	1	3	HWRPB cycle frequency (462962962) seems inaccurate - using the measured value of * Hz
L122		32	1	3	if=eth0, addr=* mask=255.255.255.0, gw=255.255.255.255,
L82		32	1	3	syslog-ng: syslog-ng version 1.4.15 starting
L147		32	1	2	P3 module inserted and registered
L81		32	1	2	registering IP recv function fffffe0000a81200, dev "myrIP0"
L149		32	1	2	myrIP module inserted
L146		32	1	1	rte-init: /cplant/init.d/enfs_client running: mount_nfs start

Conclusion:

Simple time statistics enable a first-order grouping of rigid time-correlated messages, but comparison of complete inter-arrival histograms would be more effective.



Teiresias: Convolution Phase Detail

2. “Convolution” phase: combine resulting patterns into maximal irredundant “motifs”

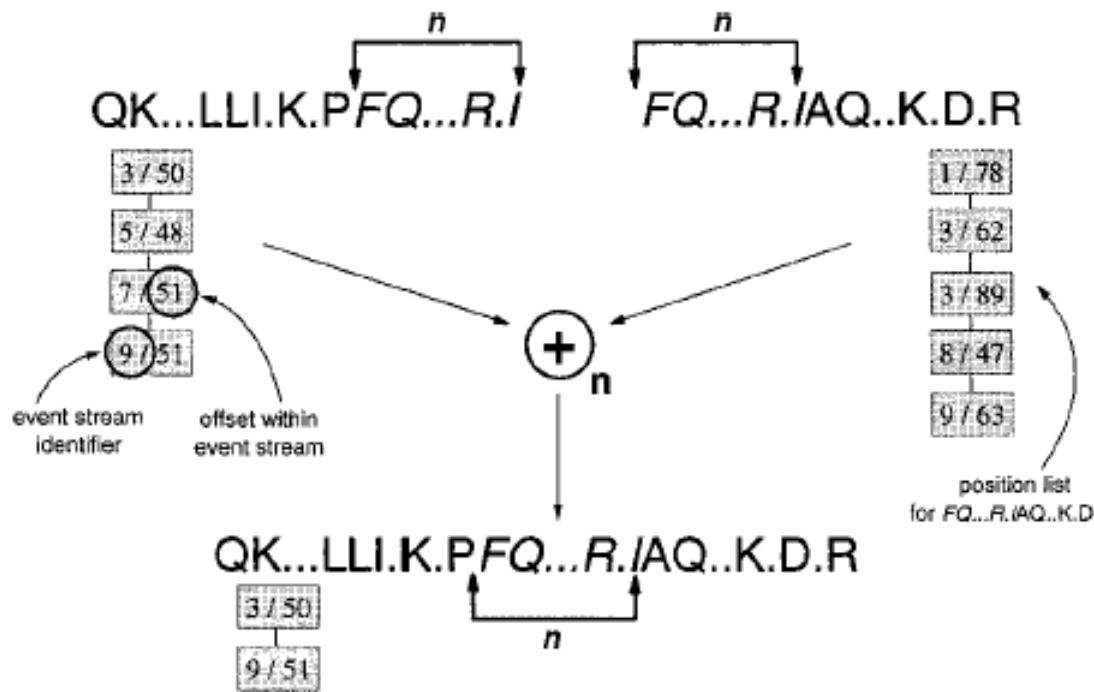


FIG. 1. The convolution phase of TEIRESIAS.

(Figure from Rigoutsos, Floratos, Parida, Gao, Platt, “The Emergence of Pattern Discovery Techniques In Computational Biology”, Journal of Metabolic Engineering, 2(3):159-177, July 2000)



slctit (SLCT enhancements)

- Common, deviant, and anomalous message types are presented distinct from each other
- Added “syslog” super-argument (-x)
- Common message types are sorted by time of first appearance (approximates sort by decreasing support (k), but also exposes time ordering)
- Message type inter-arrival median and stddev are presented (exposes periodicity and time-correlated groups)
- Interactive review via logview



Supercomputer RAS Via Informatics

