Building a 100K log/sec system

David Lang
Intuit
david@lang.hm
Talk materials available at
http://talks.lang.hm/talks/topics/Logging

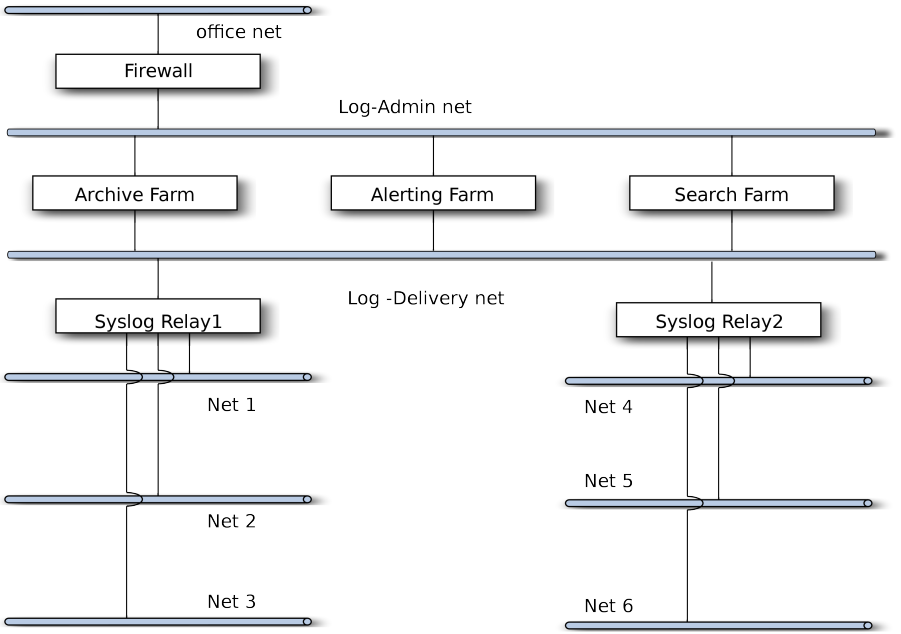
Starting Conditions

- In 2006 we had logging that had evolved
- 135 networks connected by 100 sets of firewalls
  - Proxies, no routes between networks
- 100% growth for 7 of the prior 10 years
  - Current logs estimated at ~12k logs/sec
  - 3 years of growth ~100K logs/sec

Design goals

- Gather all logs generated by any software or hardware in the company.
- Have an alerting engine that generates alerts based on individual or combinations of log messages
- Allow for rapid ad-hoc searching of the logs, both for Fraud investigations and for Troubleshooting.
- Maintain an archive of logs for many years (data retention policy set by the Legal department and driven by the need to provide logs of financial transactions to Banks)
- Generate periodic reports summarizing data in the logs
- Be able to run for at least three years without needing any architectural changes. Proactively identify what the expected bottlenecks, and produce plans to address them

Architecture



- Gathering logs
- Transporting logs
- Delivering logs
- Analyzing logs
  - Archiving
  - Alerting
  - Reporting
  - Searching

Transporting Logs

- Standards Based
- Most existing logs syslog
- Evaluated performance of syslog options
  - Selected rsyslog
    - Performance
    - Flexibility
- “Fixing” logs during transport
  - Example: Cisco

    <pri>timestamp IP tag: message

    <pri>timestamp NAME : tag: message

Gathering Logs

- Syslog
- Logger
- Rsyslog imfile
- Custom scripts

Delivering Logs

- 100K logs/sec
- average of ~250 Bytes/log
- Four (or more) destinations for each log message
- Gig-E wire speed 125MB/sec (theoretical)

Gig-E likely to be a limiting factor

Solution: Multicast MAC (CLUSTERIP on linux)

Multicast MAC Intro

- Part of Ethernet spec, NOT IP multicast
  - Transparent to applications, firewalls, routers
- Configure in IPTABLES
- Designed for load balancing
- Sends one copy of the packet
  - Received by multiple machines
  - Processed by one machine

Multicast MAC Example

```
/sbin/iptables -I INPUT -d 192.168.1.5 -i eth0 \
 -j CLUSTERIP  --new \
--clustermac 01:02:03:04:05:06 \
--total-nodes 3 --local-node 1 \
--hashmode sourceip-sourceport
```

Multicast MAC

1. Source 192.168.1.1 port 1025 Dest 192.168.1.5 port 514

   hashes to 13 13%3 = node 1

2. Source 192.168.1.1 port 1026 Dest 192.168.1.5 port 514

   hashes to 14 14%3 = node 2

3. Source 192.168.1.1 port 1027 Dest 192.168.1.5 port 514

   hashes to 15 15%3 = node 3

Multiple Recipients

- With connectionless protocols you can have more than one machine process a packet
- Configure two machines to receive all packets

  /sbin/iptables -I INPUT -d 192.168.1.5 -i eth0 -j CLUSTERIP –new \

  –clustermac 01:02:03:04:05:06 –total-nodes 1 –local-node 1 \

  –hashmode sourceip
- This also works with one system being 1 of 4 and another being 1 of 1

Multicast MAC pros/cons

Pros

- Speed limit now receiving port speed
- Add/Remove clusters without configuring senders
- Add/Remove members of clusters without configuring senders

Cons

- UDP only
  - Log length limited to packet size
  - “unreliable” delivery
- Must use Linux/BSD for receiving systems

Analyzing Logs: Archiving

- Rsyslog writing to traditional flat files rotated every minute.
- Logs split by rsyslog into type categories
- File rotation into directories year/month/day/type-messages.HHMM

Analyzing Logs: Alerting

- Initially two implementations
  - Nitro Security Appliances (commercial)
    - Could not handle load
    - Could not handle syslog relays
  - Simple Event Correlator (Open Source)
    - Works well, but single threaded.
    - See 2010 LISA paper by Paul Krizak “Log Analysis and Event Correlation Using Variable Temporal Event Correlator (VTEC)” for a great concept for scaling.
      - Split logs by type, have a single process look for things within that type of logs, and when it finds things, generate log messages back into the system.
      - Have a “Master” process just looking for log messages from the sub-processes and alert on combinations of those

Analyzing Logs: Reporting

- Artificial Ignorance
  - Filter out but count messages that are known to be uninteresting
  - Split logs that you recognize off to separate scripts to summarize them
  - Sort the remaining messages by the most common messages (filter out minor differences)
- Report each hour on that hour's logs
- Report each day on that day's logs

Analyzing Logs: Reporting

- Reporting scripts run on Archive server
- Summary scripts are Bash, Awk, Perl
  - Start simple and optimize/re-write as needed
- Example of final summary command

```
|cut -c 17- |sed s/"port [0-9]* "/"port PORT "/g
|sed s/\[[0-9]*\]/"[PID]"/g|sed s/"pid=[0-9]*"/pid=PID/g
|sed s/"FIFO threshold to [0-9]* bytes"/"FIFO threshold to BYTES bytes"/g
|sort -S 2G -T $TMPDIR |uniq -c |sort -S 2G -T $TMPDIR -rn
>other-logs
```

Analyzing Logs: Searching

- Investigated SenSage, Splunk, and homebrew with Greenplum (Postgres derived cluster DB)
  - Sensage: Very Expensive, no front-end
  - Splunk: Expensive, nice front-end
  - Greenplum: Cheap, no front-end

Analyzing Logs: Searching

- Initial Splunk cluster by-the-book with 4 indexer systems plus two forwarder systems
- Each indexer

  2x 2 core CPU, 16G ram, RAID 1 2x300G 15K SCSI + RAID 10 10x1TB SATA

- After one year, a query could take 15 minutes

Analyzing Logs: Searching

- After testing many things, re-architected Splunk
- Eliminated forwarders
- Went to a raid 10 set of forwarders
  - Distributed load across 10 pairs of indexers
- Upgraded indexers
  - 2x 4 core CPU, 64G ram, X25E SSD + RAID 6 16x1TB SATA

Analyzing Logs: Searching