

Log aggregation and analysis

Common logging issues

- Not centralised
- Centralised logging over UDP
- Not searchable
- Not verbose enough
- Too verbose

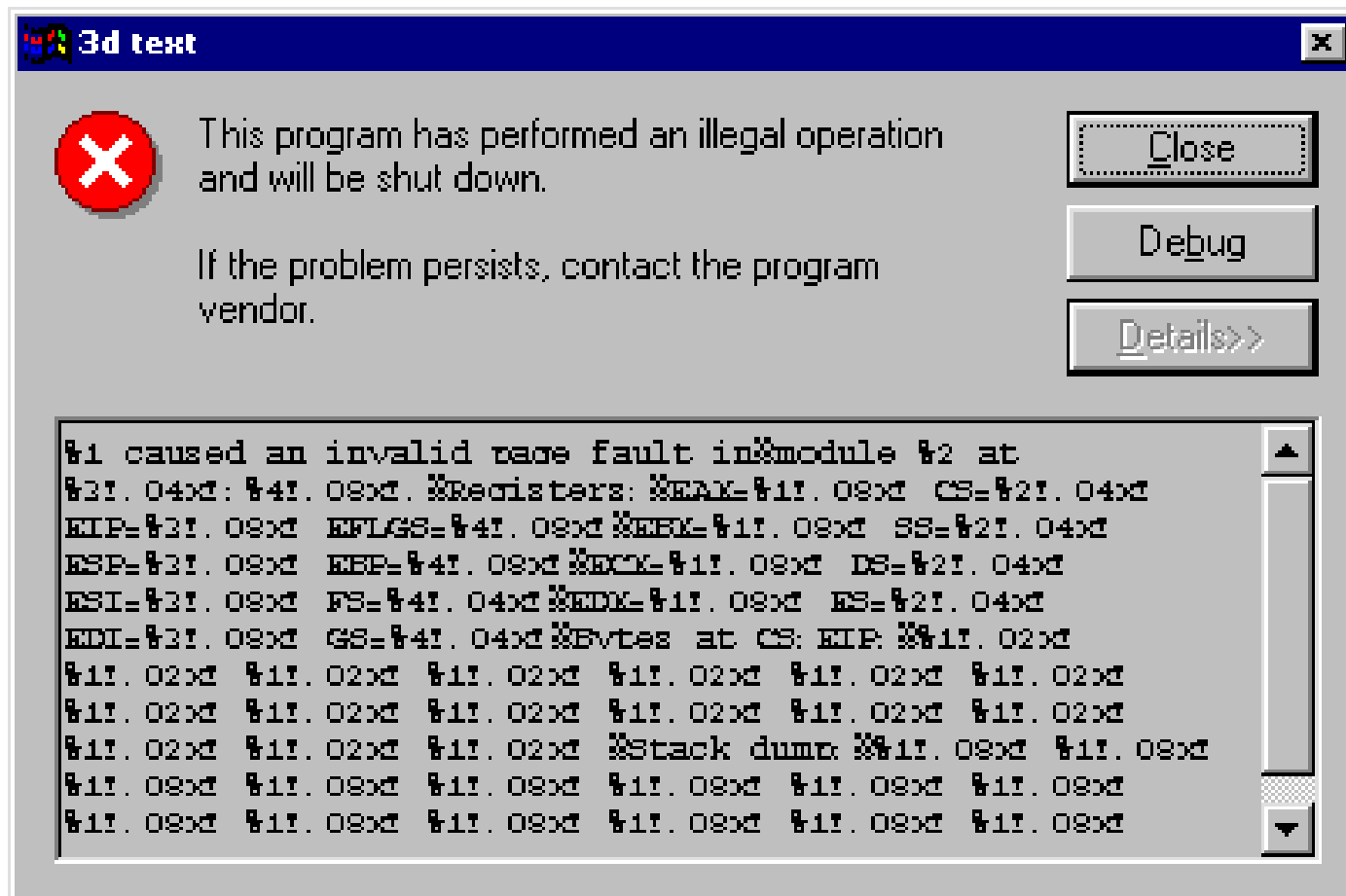
Not centralised logs?

- Web servers
- Java application servers
- “Modern” applications which do not use syslog
- Cron scripts

Not searchable

- Logs on disk can only be searched by people with access to the system
- Humans needing access to logs:
 - Administrators (sysadmin, netadmin, DBA)
 - Developers
 - Customer support
 - Business analysts

Not verbose enough



Too verbose

```
java.lang.RuntimeException
  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
  at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:39)
  at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:27)
  at java.lang.reflect.Constructor.newInstance(Constructor.java:513)
  at org.codehaus.groovy.reflection.CachedConstructor.invoke(CachedConstructor.java:77)
  at org.codehaus.groovy.runtime.callsite.ConstructorSite$ConstructorSiteNoUnwrapNoCoerce.callConstructor(ConstructorSite)
  at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCallConstructor(CallSiteArray.java:52)
  at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:192)
  at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callConstructor(AbstractCallSite.java:196)
  at newifyTransform$_run_closure1.doCall(newifyTransform.gdsl:21)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
  at java.lang.reflect.Method.invoke(Method.java:597)
  at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:86)
  at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
  at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
  at groovy.lang.MetaClassImpl.invokeMethod(MetaClassImpl.java:893)
  at org.codehaus.groovy.runtime.callsite.PogoMetaClassSite.callCurrent(PogoMetaClassSite.java:66)
  at org.codehaus.groovy.runtime.callsite.AbstractCallSite.callCurrent(AbstractCallSite.java:151)
  at newifyTransform$_run_closure1.doCall(newifyTransform.gdsl)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
  at java.lang.reflect.Method.invoke(Method.java:597)
  at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:86)
  at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
  at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
  at groovy.lang.MetaClassImpl.invokeMethod(MetaClassImpl.java:893)
  at org.codehaus.groovy.runtime.callsite.PogoMetaClassSite.call(PogoMetaClassSite.java:39)
  at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:121)
  at org.jetbrains.plugins.groovy.dsl.GroovyDslExecutor$_processVariants_closure1.doCall(GroovyDslExecutor.groovy:54)
  at sun.reflect.GeneratedMethodAccessor61.invoke(Unknown Source)
  at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
  at java.lang.reflect.Method.invoke(Method.java:597)
  at org.codehaus.groovy.reflection.CachedMethod.invoke(CachedMethod.java:86)
  at groovy.lang.MetaMethod.doMethodInvoke(MetaMethod.java:234)
  at org.codehaus.groovy.runtime.metaclass.ClosureMetaClass.invokeMethod(ClosureMetaClass.java:272)
```

Logging over UDP

- Lossy
 - You lose logs when you need them the most
 - When things go wrong and you need to debug
 - Network overload
 - Packet drops
 - UDP fragmentation is a problem
 - Must be handled by the application
 - Many popular applications do not (node.js for example)

Access control

- Limiting access to centralised logging is another issue
 - Especially to sensitive data
 - Debug logs
 - Personally identifiable information
- Corporate policies
 - Helpdesk access to email logs?
 - Netflow logs?

The ELK stack

- “New” hotness
 - Elasticsearch
 - Logstash
 - Kibana
- Good for basic textual logging
- Somewhat fragile under very high load

Elasticsearch

- A full text search based on Lucene
 - Java application (Use a new JRE – 1.8 is good)
- Limited indexing features
 - Indexing options could be better
- High performance
- Scalable

Logstash

- Jruby based log processor
- Ships with input and output filters for a very wide variety of input
 - Syslog
 - Email
 - Apache logs
 - ...
 - Ability to write custom filters

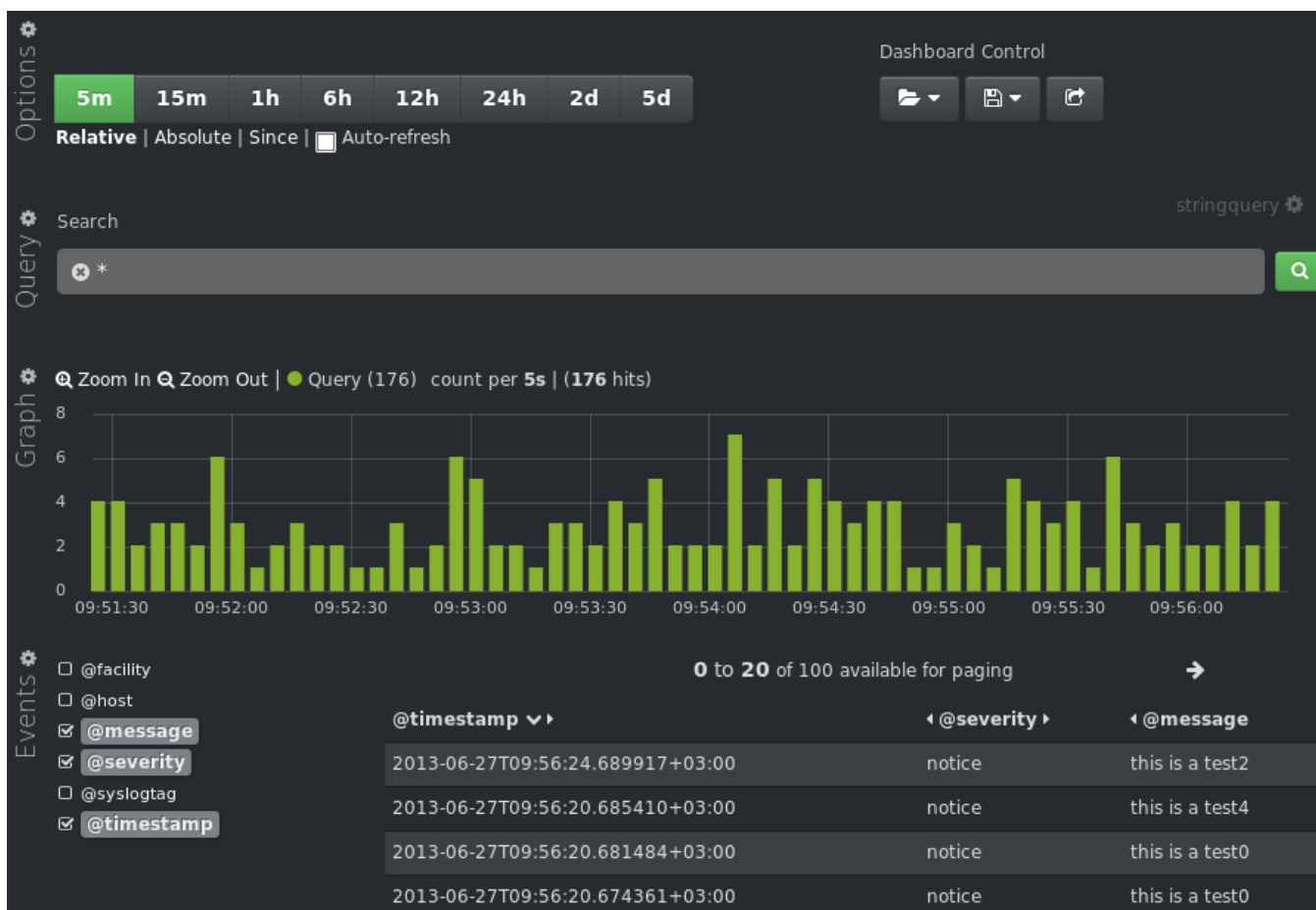
Kibana

- Javascript based frontend for search
- Shows raw logs and some graphing based on number of events
- Ability to store predefined dashboards
- AAA is delegated to the webserver

Benefits

- Store indexed logs
 - Quickly search through logs
 - Saves admin time on debugging tickets
 - ES will duplicate indices, avoid wasting RAID space
- Limit access to logs by index name
 - Alias indices to save on space
- Elasticsearch can expire and delete indices
 - `elasticsearch-curator`

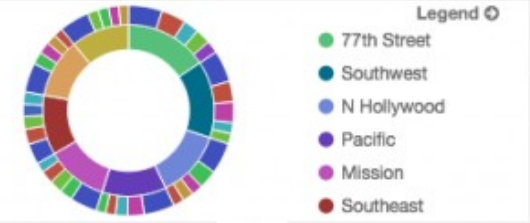
Default Kibana output



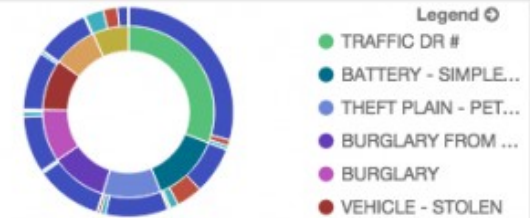
LAPD: Tile map



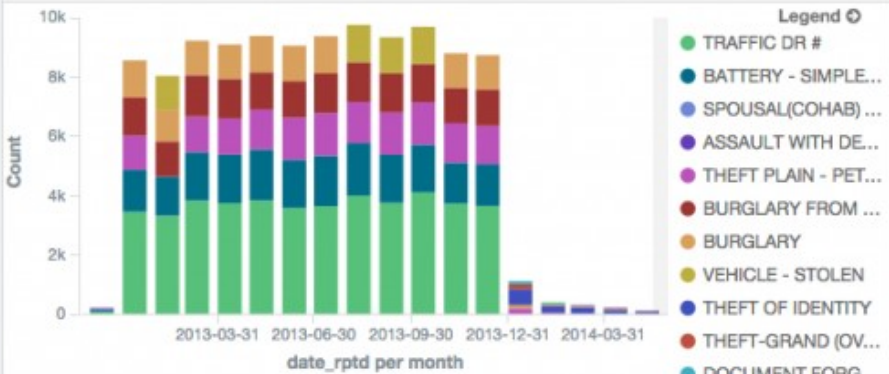
LAPD: Crime Type by Area



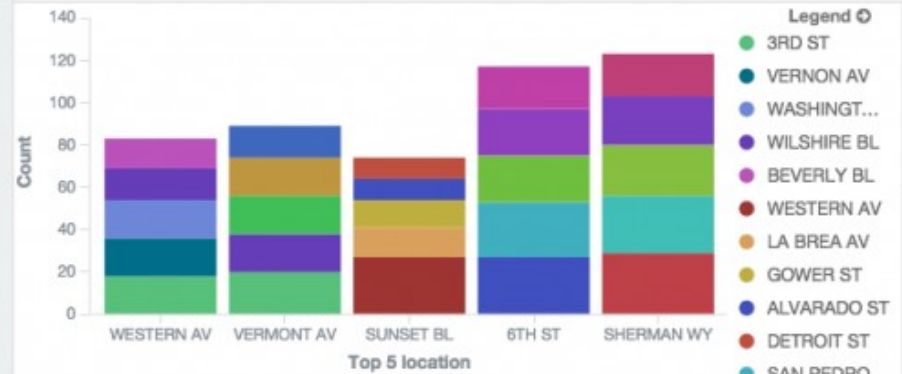
LAPD: Crime Status by Crime Type



LAPD: Monthly histogram by Crime Type



LAPD: Intersections



Limitations

- ES does not have good query rate limiting
 - ES will try to execute all queries and run out of resources
 - No automatic query killer yet
 - Elasticsearch doesn't have good failure recovery semantics
 - A hard reboot of a working cluster can take hours
- Query logging is lacking
 - Kibana queries are logged by the webserver
 - Direct access to ES gives no query logs

