ApacheCon NA 2013
Portland, Oregon
February 26th – 28th, 2013

Monitoring Apache Tomcat and the Apache Web Server

Rainer Jung

Agenda

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

Monitoring Goals

- Failure Detection, Red/Green Status, Alarms, Notifications
  - Automatic detection and notification of failures
  - No false positives
- Mostly only understood and implemented for platform basics and end-to-end
  - File system free percentage, CPU % busy
  - Application login, success of test transactions
- Only notify about root cause
  - Hard to fulfill

Motivation – Monitoring

- Further Monitoring Goals
  - Continuous collecting of runtime metrics
    - Polling
    - Storing
    - Accumulating and visualizing
  - Uses:
    - Problem root cause analysis
      - Which metrics are exactly needed not known in advance
    - Capacity management
      - Do we need to add resources
      - Do we need to adjust software sizing/configuration

Motivation – Categories of Monitoring Data

- What kind of metrics are we looking for?
  - Application load and response times
  - Utilization of software components
    - Pools, caches
  - Resource usage
  - Java memory and GC behavior
- This talk is about metrics readily available in httpd and Tomcat, not about
  - End to end monitoring, Application specific monitoring, log file monitoring

Motivation – Non-Monitoring Data

- What we can't get from monitoring
  - Is the application waiting on another system (back end, database, …)?
  - Is the application waiting to acquire locks (lock contention, software design problem)?
  - Are we looping in code, or example due to an unexpected error condition?
- For this we would need to analyze thread dumps
  - That would be another talk

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

Java Management Extensions

- Java Management Extensions (JMX)
  - A standard in the Java world
  - Can be used to expose internal application states
    - Sizes (pools etc.), counters (requests, errors, etc.)
    - Configuration settings
    - Structured and nested data supported
  - Operations supported
    - Examples: reset, resize, change log level, ...
  - Notifications (Emitter, Listener)
    - Events, for example threshold alarms

Java Management Extensions – MBeans

- MBeans
  - Information is grouped in MBeans (Management Beans)
  - MBeans have a structured name (the ObjectName), a list of attributes (the actual data) and sometimes operations
  - Attributes are often scalar, can also be nested structures
  - Simple example:

```
Name: Catalina:type=ThreadPool,name="http-bio-8080"
     ^^Domain^^
currentThreadCount: 10
currentThreadsBusy: 1
maxThreads: 200
connectionCount: 2
```

MBeans and the MBeanServer(s)

- MBeans are registered via their name.
  The registry is called an MBeanServer
- MBean attributes can be retrieved via (remote) access to the MBeanServer using their name
- There's a common base set of MBeans available in each JVM (not only Tomcat), named Platform MBeans
- Tomcat (and most other containers) provides lots of additional more specific MBeans
- It is easy to provide application specific MBeans.
  Developers should embrace this!

Java Management Extensions – Remote JMX Access

- Remote JMX Access to an MBeanServer
  - Needs a JMX client, typically a Java client
  - Some system properties must be set
    - http://docs.oracle.com/javase/6/docs/technotes/guides/management/agent.html#gdevf
    - -Dcom.sun.management.jmxremote \
      -Dcom.sun.management.jmxremote.port=9876 ...
  - Caution: for production systems always configure access control
  - Trouble with firewalls (RMI which opens additional ports)
    - Solution for Tomcat: JmxRemoteLifecycleListener
      See: http://tomcat.apache.org/tomcat-7.0-doc/config/listeners.html

Java Management Extensions – JMX Clients

- JMX-Clients (few examples)
  - JVisualVM
    - Comes with the Oracle JDK
    - Not an enterprise tool:
      - Useful for ad hoc inspecting MBeans
      - Only GUI mode
      - No way to continually persist data
  - jmxterm (interactive JMX shell)
- Most (all) monitoring solutions offer JMX integration

Java Management Extensions – VisualVM Demo

- JVisualVM Demo
  - JVisualVM connected to Sleep.java
  - JVisualVM connected to JVisualVM
  - JVisualVM connected to Tomcat

Java Management Extensions – Client Problems and Alternatives

- Problems with JMX-Clients
  - Creating a new JVM process for each monitoring poll is a killer on the polling monitoring server
  - So use a persistent JMX client
- Alternative: use another protocol
  - Run an agent in the JVM, that can be reached by a non-Java technology/protocol. Example: HTTP
  - Or run a proxy written in Java that talks JMX to the target but can be queried for example using HTTP

- Motivation
- Java Management Extensions (JMX)
- **Some Remarks**
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

Some Remarks

- Measurements must be taken automatically and be persisted
  - Interactive observations (JVisualVM) are not sufficient
    - No history, each user polls separately
    - Can be useful to get an idea what to track
  - Typical poll interval 1 minute
  - In addition to polling and persisting the data
    - We need to think about thresholds
      - For which measurements are they adequate, which values to choose
    - We need to automatically visualize the data
    - Access to raw data should be available if needed

Some Remarks

- Hurdles
  - Scalar attributes versus MXBeans (OpenMBeans)
    - More and more MBeans provide structured and nested data (MXBeans)
    - Tools could automatically inspect the structure of the data
    - Some tools still do not support that
    - Check your tool support before implementing your own MBeans as MXBeans

Some Remarks

- More Hurdles
  - MBeans typically reflect source code structure
    - To many MBeans, to hard to understand
      - Not always optimal granularity
      - Many MBeans of the same type
      - Might lead to unacceptable polling load if tool does not support appropriate bulk requests
  - Necessity to use simple mathematical operators
    - Not always the right level of information in the MBeans
      - Example: maximum and current pool size, but not current percentage
    - Need to deduce deltas or rates from counters, quotients of metrics, quotients of deltas

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

Monitoring Apache Tomcat – Getting the Data

- Let's use HTTP to retrieve JMX data from Tomcat
  - JMXProxy is part of the Tomcat Manager webapp
    - https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html#Using_the_JMX_Proxy_Servlet
    - Role “manager-jmx” needed
    - Demo
  - Jolokia (http://www.jolokia.org/)
    - For Tomcat: deployable webapp
    - REST style interface, data delivered in JSON, bulk request support
    - Clients Jmx4Perl, j4psh, Javascript client API
  - Or write your own servlet (see code of JMXProxy)

Monitoring Apache Tomcat – Plattform MBeans

- Starting with platform MBeans
  - These are available in every JVM, not just Tomcat

Monitoring Apache Tomcat – Plattform MBeans

- OperatingSystem

```
Name: java.lang:type=OperatingSystem
AvailableProcessors: 4
ProcessCpuTime: 1445000000
                 ^^^nano seconds^^^
SystemLoadAverage: 0.14453125
TotalPhysicalMemorySize: 4080713728
TotalSwapSpaceSize: 8877690880
FreePhysicalMemorySize: 911835136
FreeSwapSpaceSize: 3356606464
CommittedVirtualMemorySize: 232214528
OpenFileDescriptorCount: 31
MaxFileDescriptorCount: 65536
```

Monitoring Apache Tomcat – Plattform MBeans

- OperatingSystem
  - ProcessCpuTime: Rate gives "CPU seconds per second". Approximation of needed CPU concurrency (CPU cores used). Threshold applies
  - SystemLoadAverage: usual 1 minute average. Threshold.
  - CommittedVirtualMemorySize: includes native memory, for example used by JNI. Threshold applies, for example in case of native memory leaks
  - Open/MaxFileDescriptorCount: useful to check FD limits. Quotient gives percentage used of allowed FDs

Monitoring Apache Tomcat – Plattform MBeans

- Runtime

```
Name: java.lang:type=Runtime
Uptime: 25745
```

- Threading

```
Name: java.lang:type=Threading
TotalStartedThreadCount: 24
PeakThreadCount: 23
ThreadCount: 22
DaemonThreadCount: 21
```

  - TotalStartedThreadCount: Rate gives "threads started per second"
  - ThreadCount or DaemonThreadCount: threshold applies

Monitoring Apache Tomcat – Plattform MBeans

- ClassLoading

```
Name: java.lang:type=ClassLoading
LoadedClassCount: 3058
TotalLoadedClassCount: 3058
UnloadedClassCount: 0
```

- Compilation

```
Name: java.lang:type=Compilation
TotalCompilationTime: 5270
```

- Both typically not very interesting
  - Possibly useful when running other languages in the JVM
  - Somewhat useful when comparing releases

Monitoring Apache Tomcat – Plattform MBeans

- Memory

```
Name: java.lang:type=Memory
ObjectPendingFinalizationCount: 0
HeapMemoryUsage:    committed=109117440,
   init=63761152, max=907870208, used=34542904
NonHeapMemoryUsage: committed=25100288,
   init=24313856, max=136314880, used=23915824
```

  - Not interesting, because several memory regions are thrown together
  - But see next MBean type ...

Monitoring Apache Tomcat – Plattform MBeans

- MemoryPool
  - Data available for: Eden, Survivor, Tenured, Perm and Code Cache
  - Example for Eden

```
Name: java.lang:type=MemoryPool,name=PS Eden Space
Usage:            committed=63963136,
        init=15990784, max=335151104, used=14094200
PeakUsage:        committed=63963136,
        init=15990784, max=335151104, used=31981568
CollectionUsage: committed=63963136,
        init=15990784, max=335151104, used=0
```

  - CollectionUsage: better use next MBean

Monitoring Apache Tomcat – Plattform MBeans

- GarbageCollector

```
Name: java.lang:type=GarbageCollector,name=PS Scavenge
CollectionCount: 4
CollectionTime: 127
LastGcInfo: GcThreadCount=4, id=4,
      startTime=10275, endTime=10302, duration=27,
      memoryUsageBeforeGc={...}, memoryUsageAfterGc={}
```

  - Data Available for eden GC (example: “PS Scavenge”) and for tenured GC (example: “PS MarkSweep”)
  - CollectionTime: cumulated duration in milliseconds
  - startTime, endTime: milliseconds since JVM start

Monitoring Apache Tomcat – Plattform MBeans

- GarbageCollector
  - Example "tenured" cleaning in "PS MarkSweep"
    - memoryUsageBeforeGc:

```
[PS Old Gen]={committed=42532864, init=42532864,
max=680919040, used=19563280}
```

    - memoryUsageAfterGc:

```
[PS Old Gen]={committed=42532864, init=42532864,
max=680919040, used=18396688}
```

Monitoring Apache Tomcat – Global Tomcat MBeans

- Now Tomcat MBeans:
  First the global Tomcat MBeans
- ThreadPool (for each Connector in server.xml)

```
Name: Catalina:type=ThreadPool,name="http-bio-8080"
currentThreadCount: 10
currentThreadsBusy: 1
maxThreads: 200
connectionCount: 2
```

  - currentThreadsBusy/maxThreads: thread pool usage
    - What is "busy"? Depending on connector: handles a connection (BIO) or handles a request (NIO and APR)
  - connectionCount: off by 1, useful when using NIO or APR

Monitoring Apache Tomcat – Global Tomcat MBeans

- GlobalRequestProcessor (for each Connector)

```
Name: Catalina:type=GlobalRequestProcessor,
               name="http-bio-8080"
requestCount: 20
errorCount: 1
processingTime: 3766
maxTime: 1902
bytesReceived: 0
bytesSent: 615691
```

  - requestCount: Rate gives throughput (requests per sec)
    - but: what is an error?
  - delta(errorCount)/delta(requestCount): error rate

Monitoring Apache Tomcat – Global Tomcat MBeans

- GlobalRequestProcessor continued
  - processingTime: cumulated in milliseconds.
    Rate is average request concurrency in last interval (!)
  - delta(processingTime)/delta(requestCount): average request processing time in the last interval (!)
  - bytesReceived, bytesSent: do not contain headers.
    Rate is approximation for bandwidth (headers missing).
    delta(bytes)/delta/requestCount) is average body size in the last interval

Monitoring Apache Tomcat – Global Tomcat MBeans

- RequestProcessor (roughly for each thread)

```
Name: Catalina:type=RequestProcessor,
worker="http-bio-8080",name=HttpRequest6
requestCount: 3
errorCount: 0
...
remoteAddr: 0:0:0:0:0:0:0:1
currentUri: /sleep.jsp
requestProcessingTime: 9432
```

  - currentUri: attribute only present if the processor currently works on a request
  - requestProcessingTime: detect long running requests on the fly

Monitoring Apache Tomcat – Global Tomcat MBeans

- DataSource (Tomcat provided database conn. pools)

```
Name: Catalina:type=DataSource,
      class=javax.sql.DataSource,
      name="jdbc/myappDB"
numIdle: 3
numActive: 1
maxActive: 7
```

  - maxActive: configuration item
  - 100*numActive/maxActive: current pool use in percent
  - Can be per webapp if defined in the webapp, ObjectName then slightly different

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- Finally the per Webapp Tomcat MBeans
- Manager (Session Management)

```
Name: Catalina:type=Manager,context=/,
               host=localhost
sessionCounter: 1234
activeSessions: 19
expiredSessions: 1214
```

  - sessionCounter, expiredSessions: sessions created resp. expired since Tomcat start. Rate is session per second, for example "login rate" and "logout/timeout rate". Use threshold.
  - activeSessions: current number of sessions. Use threshold

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- Manager continued

```
Name: Catalina:type=Manager,context=/,
               host=localhost
maxActive: 114
rejectedSessions: 0
duplicates: 0
```

  - maxActive: can be reset without restart
  - rejectedSessions: how often was a configured maxActiveSessions limit reached
  - duplicates: how often was a session ID non-unique and had to be replaced (during session generation). Typically always "0"

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- Manager continued

```
Name: Catalina:type=Manager,context=/,
                host=localhost
sessionAverageAliveTime: 458
sessionMaxAliveTime: 1861
processingTime: 3
```

  - "alive" times: in seconds measured from session creation to expiration (be it logout or session idle timeout)
  - processingTime: cumulated elapsed milliseconds needed for session expiration handling. Usually not interesting, but could be when using a custom HttpSessionListener

- Manager continued

```
Name: Catalina:type=Manager,context=/,
                host=localhost
sessionCreateRate: 1
sessionExpireRate: 1
```

- Rate: Rates per minute for the last 100 session created/expired

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- Servlet

```
Name: Catalina:j2eeType=Servlet,name=JMXProxy,
      WebModule=//localhost/manager,
      J2EEApplication=none,J2EEServer=none
requestCount: 6
errorCount: 1
processingTime: 1360
minTime: 92
maxTime: 789
```

  - Times in milliseconds. Earlier remarks apply, see GlobalRequestProcessor

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- "default" Servlet (Tomcat provided)

```
Name: Catalina:j2eeType=Servlet,name=default,
      WebModule=//localhost/,
      J2EEApplication=none,J2EEServer=none
requestCount: 113
errorCount: 0
processingTime: 243
minTime: 0
maxTime: 6
```

  - Handles serving of static content

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- "JSP" Servlet (Tomcat provided)

```
Name: Catalina:j2eeType=Servlet,name=jsp,
      WebModule=//localhost/,
      J2EEApplication=none,J2EEServer=none
requestCount: 5
errorCount: 0
processingTime: 3312
minTime: 494
maxTime: 912
```

  - Tracks JSP execution

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- JSPMonitor

```
Name: Catalina:type=JspMonitor,name=jsp,
      WebModule=//localhost/,
      J2EEApplication=none,J2EEServer=none
jspCount: 31
jspUnloadCount: 0
jspReloadCount: 31
jspQueueLength: -1
```

  - Tracks JSP loading, reloading and unloading
  - For “unloading” see “maxLoadedJsps” on http://tomcat.apache.org/tomcat-7.0-doc/jasper-howto.html

Monitoring Apache Tomcat – Webapp Tomcat MBeans

- WebModule (Webapp)

```
Name: Catalina:j2eeType=WebModule,
      name=//localhost/manager,
      J2EEApplication=none,J2EEServer=none
processingTime: 1360
```

  - processingTime: as usual cumulated millisecond elapsed time for all handled requests (in this webapp). Sum over all servlets.
  - Unfortunately no "requestCount" available on the Webapp level (only per servlet or per connector)
  - Fixed a few minutes ago for the next TC 7 release

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- Discussion

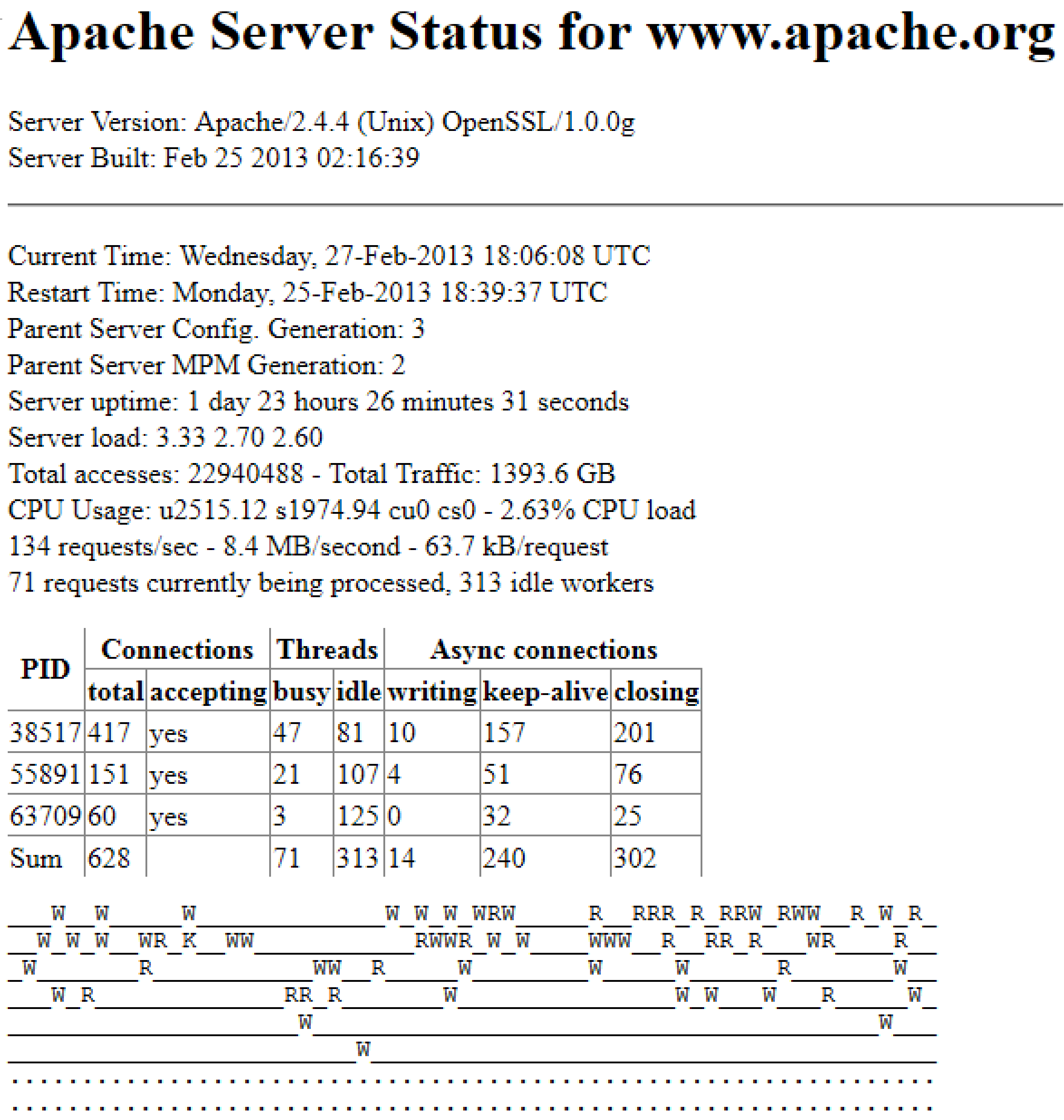
- Let's use HTTP to retrieve data from httpd
  - The module mod_status provides a web page with monitoring data

```
LoadModule status_module modules/mod_status.so
<Location /server-status>
    SetHandler server-status
    !!! YOUR ACCESS CONTROL GOES HERE !!!
</Location>
```

  - You can choose any URI you like (“/server-status”), but the “SetHandler” must be configured as-is
- Alternative not shown here: Connect to scoreboard shared memory and read data from there

Monitoring the Apache Web Server – Getting the Data

- Example:
  http://www.apache.org/server-status
  - HTML view
    - Most complete data
    - Data interesting
    - Needs to be parsed



Apache Server Status for www.apache.org

Server Version: Apache/2.4.4 (Unix) OpenSSL/1.0.0g
Server Built: Feb 25 2013 02:16:39

Current Time: Wednesday, 27-Feb-2013 18:06:08 UTC
Restart Time: Monday, 25-Feb-2013 18:39:37 UTC
Parent Server Config. Generation: 3
Parent Server MPM Generation: 2
Server uptime: 1 day 23 hours 26 minutes 31 seconds
Server load: 3.33 2.70 2.60
Total accesses: 22940488 - Total Traffic: 1393.6 GB
CPU Usage: u2515.12 s1974.94 cu0 cs0 - 2.63% CPU load
134 requests/sec - 8.4 MB/second - 63.7 kB/request
71 requests currently being processed, 313 idle workers

PID	Connections		Threads		Async connections		
	total	accepting	busy	idle	writing	keep-alive	closing
38517	417	yes	47	81	10	157	201
55891	151	yes	21	107	4	51	76
63709	60	yes	3	125	0	32	25
Sum	628		71	313	14	240	302

```
___W__W_____W_____________W_W_W_WRW_____R__RRR_R_RRW_RWW__R_W_R_
__W_W_W__WR_K__WW___________RWWR_W_W____WWW__R__RR_R___WR____R__
_W_______R___________WW__R_____W________W_____W______R_______W__
___W_R_____________RR_R_______W_______________W_W___W___R_____W_
____________________W_______________________________________W___
________________________W_______________________________________
................................................................
................................................................
```

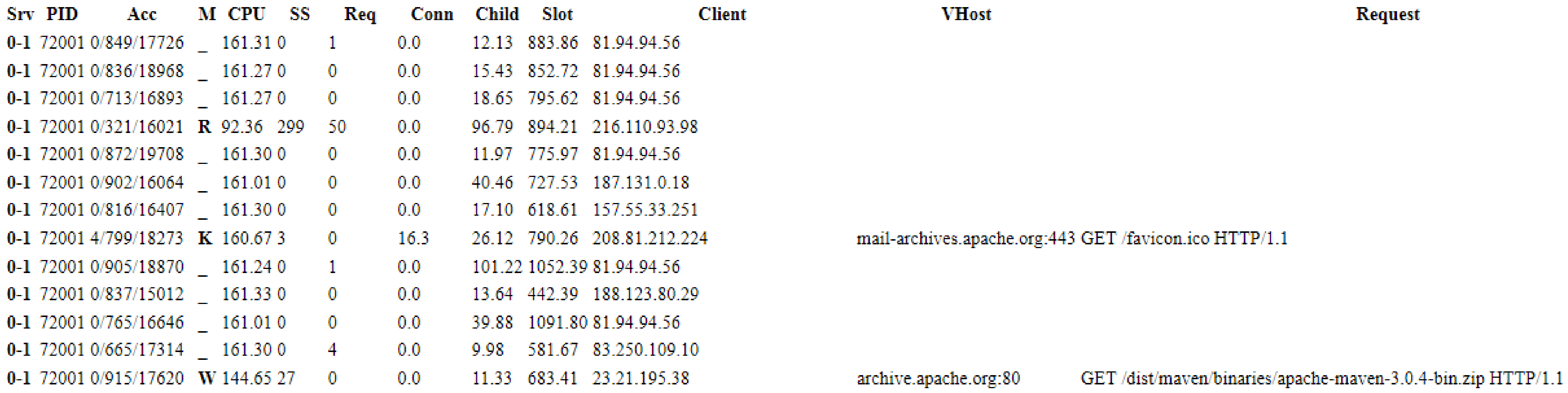
Monitoring the Apache Web Server – Getting the Data

- Example "auto":
  /server-status?auto
  - Text view
    - Incomplete data
    - Data still interesting
    - Trivial to parse

```
Total Accesses: 22984957
Total kBytes: 1463080584
CPULoad: 2.6785
Uptime: 171100
ReqPerSec: 134.336
BytesPerSec: 8756250
BytesPerReq: 65181.5
BusyWorkers: 56
IdleWorkers: 328
ConnsTotal: 577
ConnsAsyncWriting: 13
ConnsAsyncKeepAlive: 258
ConnsAsyncClosing: 250
Scoreboard: __R__W___RRR_R_R_
```

Monitoring the Apache Web Server – Getting the Data

- Per "slot" data
  - "ExtendedStatus On"
  - Recently on by default if mod_status is loaded
  - HTML table, needs to be parsed
  - Not especially interesting for monitoring

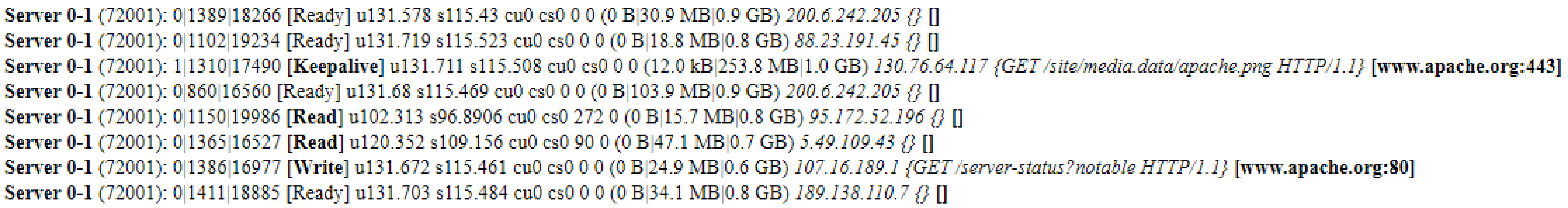


Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Client	VHost	Request
0-1	72001	0/849/17726	_	161.31	0	1	0.0	12.13	883.86	81.94.94.56		
0-1	72001	0/836/18968	_	161.27	0	0	0.0	15.43	852.72	81.94.94.56		
0-1	72001	0/713/16893	_	161.27	0	0	0.0	18.65	795.62	81.94.94.56		
0-1	72001	0/321/16021	R	92.36	299	50	0.0	96.79	894.21	216.110.93.98		
0-1	72001	0/872/19708	_	161.30	0	0	0.0	11.97	775.97	81.94.94.56		
0-1	72001	0/902/16064	_	161.01	0	0	0.0	40.46	727.53	187.131.0.18		
0-1	72001	0/816/16407	_	161.30	0	0	0.0	17.10	618.61	157.55.33.251		
0-1	72001	4/799/18273	K	160.67	3	0	16.3	26.12	790.26	208.81.212.224	mail-archives.apache.org:443	GET /favicon.ico HTTP/1.1
0-1	72001	0/905/18870	_	161.24	0	1	0.0	101.22	1052.39	81.94.94.56		
0-1	72001	0/837/15012	_	161.33	0	0	0.0	13.64	442.39	188.123.80.29		
0-1	72001	0/765/16646	_	161.01	0	0	0.0	39.88	1091.80	81.94.94.56		
0-1	72001	0/665/17314	_	161.30	0	4	0.0	9.98	581.67	83.250.109.10		
0-1	72001	0/915/17620	W	144.65	27	0	0.0	11.33	683.41	23.21.195.38	archive.apache.org:80	GET /dist/maven/binaries/apache-maven-3.0.4-bin.zip HTTP/1.1

kippdata. informationstechnologie

Monitoring the Apache Web Server – Getting the Data

- Per “slot” data – notable variant
  - /server-status?notable
  - Easier to parse
  - Still not especially interesting for monitoring



```
Server 0-1 (72001): 0|1389|18266 [Ready] u131.578 s115.43 cu0 cs0 0 0 (0 B|30.9 MB|0.9 GB) 200.6.242.205 {} []
Server 0-1 (72001): 0|1102|19234 [Ready] u131.719 s115.523 cu0 cs0 0 0 (0 B|18.8 MB|0.8 GB) 88.23.191.45 {} []
Server 0-1 (72001): 1|1310|17490 [Keepalive] u131.711 s115.508 cu0 cs0 0 0 (12.0 kB|253.8 MB|1.0 GB) 130.76.64.117 {GET /site/media.data/apache.png HTTP/1.1} [www.apache.org:443]
Server 0-1 (72001): 0|860|16560 [Ready] u131.68 s115.469 cu0 cs0 0 0 (0 B|103.9 MB|0.9 GB) 200.6.242.205 {} []
Server 0-1 (72001): 0|1150|19986 [Read] u102.313 s96.8906 cu0 cs0 272 0 (0 B|15.7 MB|0.8 GB) 95.172.52.196 {} []
Server 0-1 (72001): 0|1365|16527 [Read] u120.352 s109.156 cu0 cs0 90 0 (0 B|47.1 MB|0.7 GB) 5.49.109.43 {} []
Server 0-1 (72001): 0|1386|16977 [Write] u131.672 s115.461 cu0 cs0 0 0 (0 B|24.9 MB|0.6 GB) 107.16.189.1 {GET /server-status?notable HTTP/1.1} [www.apache.org:80]
Server 0-1 (72001): 0|1411|18885 [Ready] u131.703 s115.484 cu0 cs0 0 0 (0 B|34.1 MB|0.8 GB) 189.138.110.7 {} []
```

Monitoring the Apache Web Server – Useful Metrics

- Which metrics are useful?

```
Restart Time: Monday, 25-Feb-2013 18:39:37 UTC
Server uptime: 1 day 23 hours 53 minutes 46 seconds
```

  - Restart Time: Graceful does not count as a restart

```
Server load: 2.25 2.52 2.46
```

  - System load numbers that you would also get from the OS "uptime" command
    - Added in 2.4.4, use threshold

```
Total accesses: 23182359 - Total Traffic: 1403.6 GB
```

  - Use rate calculation for requests/second and bandwidth
  - Use quotient of delta for average size in last interval

Monitoring the Apache Web Server – Useful Metrics

- Which metrics are useful?

```
29 requests currently being processed, 227 idle
workers
```

  - Current worker thread used and available concurrency
  - Worker thread count important sizing parameter
  - Use threshold
- IMHO not as useful

```
CPU Usage: u2467.88 s1921.64 cu0 cs0 - 2.55% CPU load
134 requests/sec - 8.3 MB/second - 63.5 kB/request
```

  - CPU not reliable (child CPU handling)
  - Averages since restart

What are the busy worker threads doing?

```
__________W___________________________RR______W______
.....................................................
____R__WRWR__R_WR_____R__RRW_____R___WWR___________RW
```

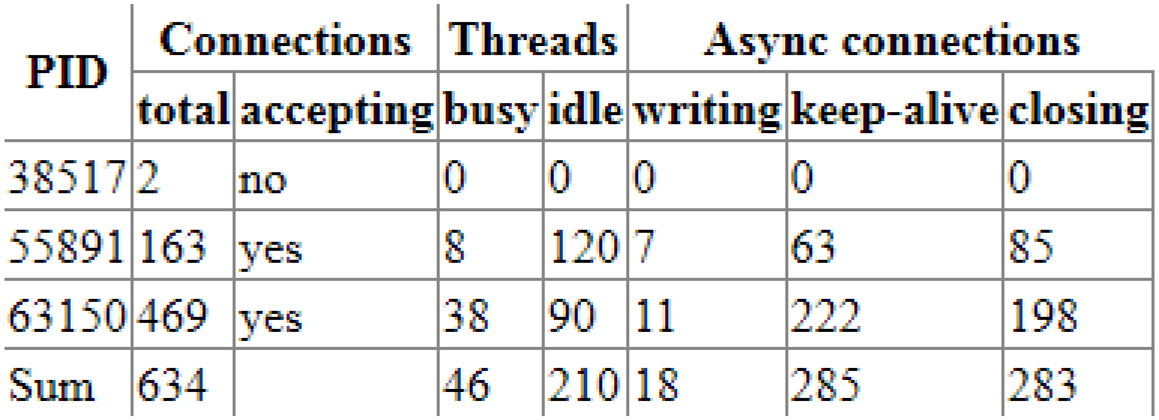
- Each character represents the status of a worker thread

```
Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"C" Closing connection, "L" Logging, "G" Gracefully finishing,
"I" Idle cleanup of worker, "." Open slot with no current process
```

- Count most important chars (W,R,K,_,.), sum up rest
- Using the event MPM most "K" will not be shown, because they no longer block a worker thread, so ...

Monitoring the Apache Web Server – Useful Metrics

- What about connections not blocking a worker thread (Event MPM)?
  - Per process and sums
  - The following typically sum up: "busy" + "writing" + "keep-alive" + "closing" = "total"
  - Only "busy" needs a worker thread
  - Monitor "Threads" and "Async" data in "Sum" row
  - All other connections states use a separate poller
    - Note how few are "busy" compared with the other states



PID	Connections		Threads		Async connections		
	total	accepting	busy	idle	writing	keep-alive	closing
38517	2	no	0	0	0	0	0
55891	163	yes	8	120	7	63	85
63150	469	yes	38	90	11	222	198
Sum	634		46	210	18	285	283

- Motivation
- Java Management Extensions (JMX)
- Some Remarks
- Monitoring Apache Tomcat
- Monitoring the Apache Web Server
- **Discussion**

Discussion

- Any questions?