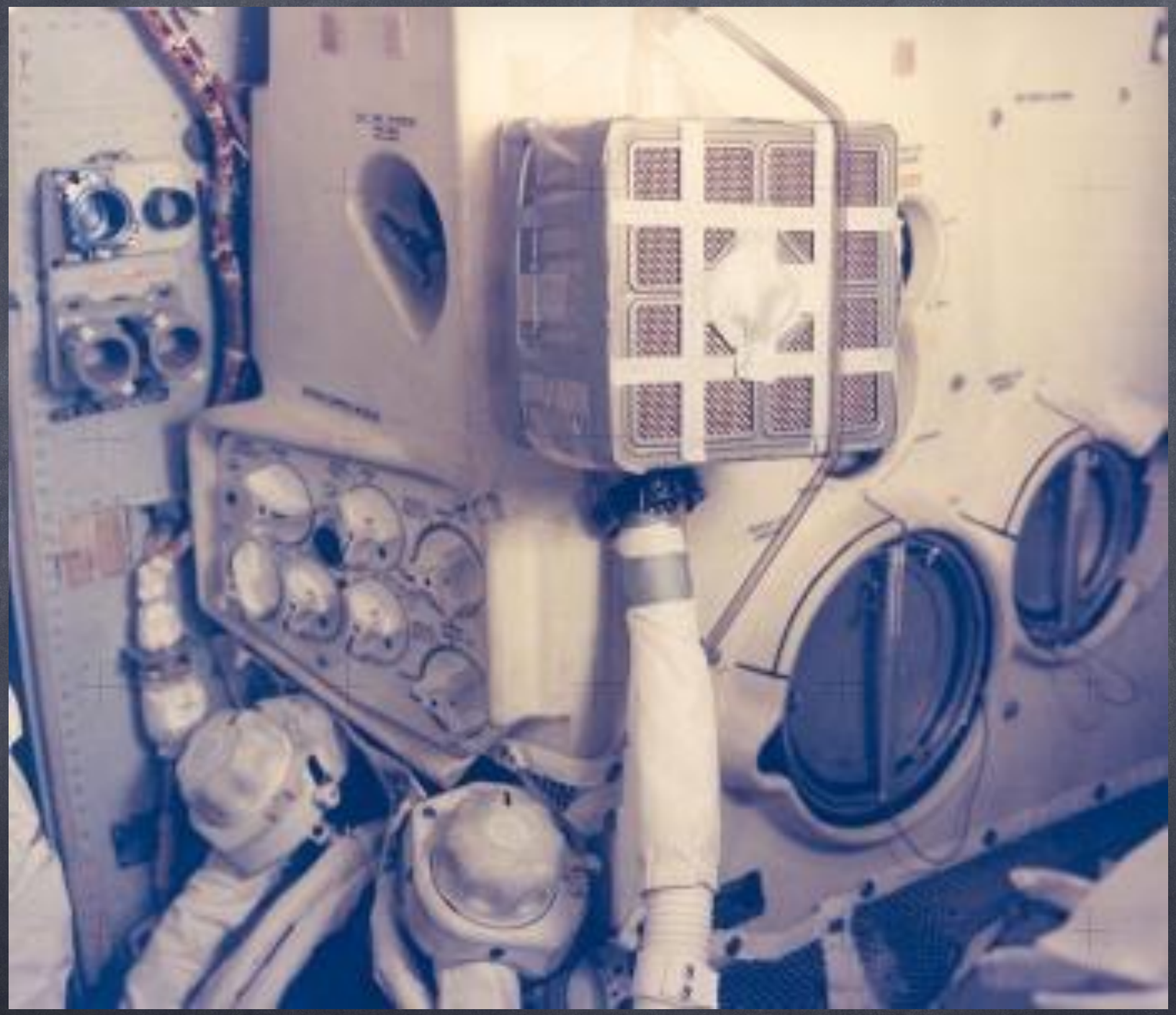


AZUL SYSTEMS

Where Do Libraries Come From?

Gil Tene, CTO & co-Founder, Azul Systems
@giltene

AZUL SYSTEMS

Mommy, Where Do Libraries Come From?

Gil Tene, CTO & co-Founder, Azul Systems
@giltene

About me: Gil Tene

- co-founder, CTO @Azul Systems
- Have been working on "think different" GC approaches since 2002
- A Long history building Virtual & Physical Machines, Operating Systems, Enterprise apps, etc...
- I also depress people by pulling the wool up from over their eyes...

* working on real-world trash compaction issues, circa 2004

AZUL SYSTEMS

Libraries

code that other people use

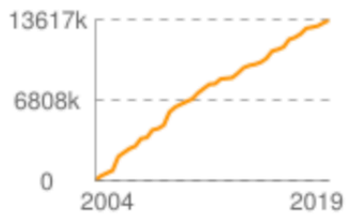
AZUL SYSTEMS

MVNREPOSITORY

HdrHistogram

Search

Indexed Artifacts (13.6M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages
- Logging Frameworks
- Logging Bridges

Home » Popular

Top Projects

1. **JUnit** — 84,813 usages — EPL
junit » junit
JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.
Last Release on Feb 2, 2019

2. **SLF4J API Module** — 34,162 usages — MIT
org.slf4j » slf4j-api
The slf4j API
Last Release on Mar 21, 2018

3. **Scala Library** — 19,759 usages — Apache
org.scala-lang » scala-library
Standard library for the Scala Programming Language
Last Release on Nov 14, 2018

4. **Guava: Google Core Libraries For Java** — 19,090 usages — Apache
com.google.guava » guava
Guava is a suite of core and expanded libraries that include utility classes, google's collections, io classes, and much much more.
Last Release on Nov 19, 2018

5. **Android AppCompat Library V7** — 17,268 usages — Apache
com.android.support » appcompat-v7
The Support Library is a static library that you can add to your Android application in order to use APIs that are either not available for older platform versions or utility APIs that aren't a part of the framework APIs. Compatible on devices running API 14 or later.
Last Release on Sep 22, 2018

6. **Apache Commons IO** — 15,712 usages — Apache
commons-io » commons-io
The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

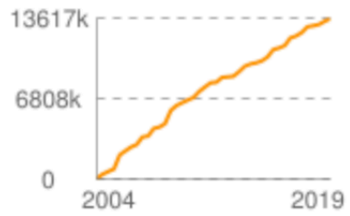
AZUL SYSTEMS

MVNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (13.6M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages

Home » Categories » Application Metrics

Application Metrics

Sort: **popular** | newest

1. **Metrics Core** — **1,308** usages
io.dropwizard.metrics » metrics-core — Apache

Metrics is a Java library which gives you unparalleled insight into what your code does in production. Metrics provides a powerful toolkit of ways to measure the behavior of critical components in your production environment.

Last Release on Dec 31, 2018

2. **Metrics Core** — **433** usages
com.codahale.metrics » metrics-core — Apache

Metrics is a Java library which gives you unparalleled insight into what your code does in production. Metrics provides a powerful toolkit of ways to measure the behavior of critical components in your production environment.

Last Release on Feb 17, 2014

3. **Metrics Core Library** — **350** usages
com.yammer.metrics » metrics-core — Apache

Metrics Core Library

Last Release on Apr 1, 2013

4. **Servo Core** — **114** usages
com.netflix.servo » servo-core — Apache

servo-core

Last Release on Jun 29, 2018

5. **SIGAR** — **91** usages
org.fusesource » sigar — Apache

SIGAR

Last Release on Mar 22, 2011

AZUL SYSTEMS

Libraries

- How do they come to be?
- Do you want to build one?
- Should you build one? Is it a good idea?
- When does it become a good idea?
- Even when it's a good idea, will it "make it"?
- What it looks like on the way to adoption
- What life looks like once its adopted
- Some rules only library writers have to live with

AZUL SYSTEMS

Libraries

How do they come to be?

AZUL SYSTEMS

Libraries

How do they come to be?

- (IMO) It never starts with "I want to build a library"
- It usually starts with you doing something
  - And then doing that same thing again
  - And again
- And then you separate that "thing" out to reuse it
  - For yourself
  - For your group
  - For you project

AZUL SYSTEMS

Libraries

How do they come to be?

- ...
- For your company...
- When it leaves the "project" it starts being a library
  - Something separate, uncoordinated, is actually using it
  - Thats when the "fun" starts
- Usually there is a "rationalization" step
  - When you realize it is being used as a library
- Things get re-stated in more generic terms

AZUL SYSTEMS

Example: HdrHistogram

- HdrHistogram is… a histogram
  - How much can you write about a histogram?
  - It turns out: a lot
- Initial "Library" implementation written in Java
  - With language implementations in C, C#, Javascript, Python, Rust, Go, Erlang, Haskel, …
  - Used by SpringBoot, Hystrix, Elasticsearch, Log4j2, Storm, Gatling, Akka, Cassandra, Aeron, …, …
- So what is it?

AZUL SYSTEMS

Example: HdrHistogram

what is it?

- Histograms count things in buckets
  - Thats simple, seemingly
  - Which buckets? What sizes? how many?
- Common, simple, in-line implementations:
  - Linear buckets e.g. [0-99][100-199][200-299]...
    - Need 3.6B buckets to cover 1 usec to 1 hour
  - Logarithmic e.g. [0-1][2-3][4-7][8-15][16-31][31-64]...
    - Only need ~32 to cover 1usec to 1 hour
    - But can't tell the difference between 2 seconds and 3.9 seconds

HdrHistogram

what is it?

HdrHistogram: A High Dynamic Range Histogram.

A Histogram that supports recording and analyzing sampled data value counts across a configurable integer value range with configurable value precision within the range. Value precision is expressed as the number of significant digits in the value recording, and provides control over value quantization behavior across the value range and the subsequent value resolution at any given level.

For example, a **Histogram** could be configured to track the counts of observed integer values between 0 and 3,600,000,000,000 while maintaining a value precision of 3 significant digits across that range. Value quantization within the range will thus be no larger than 1/1,000th (or 0.1%) of any value. This example Histogram could be used to track and analyze the counts of observed response times ranging between 1 nanosecond and 1 hour in magnitude, while maintaining a value resolution of 1 nanosecond up to 1 microsecond, a resolution of 1 microsecond up to 1 millisecond, a resolution of 1 millisecond (or better) up to one second, and a resolution of 1 second (or better) up to 1,000 seconds. At it's maximum tracked value (1 hour), it would still maintain a resolution of 3.6 seconds.

AZUL SYSTEMS

HdrHistogram

what is it?

- Example: track values from 1 nanosecond to 1 hour
  - With a precision of 3 decimal points
  - Histogram = new Histogram(3600000000000L, 3);
- Above (example) maintain 3 decimal point precision
  - 1 usec resolution up to 1 millisecond
  - 1 millesecond resolution up to 1 second
  - 1 second resolution up to 1000 seconds
  - no worse than +/- 3.6 seconds up to 1 hour

AZUL SYSTEMS

HdrHistogram

what is it?

- It is FAST (>200M recording per second on a laptop)
  - usually faster than measuring time
- It is zero-allocation
- It has a wire format
  - compact, and works across languages
- It has convenience things
  - Recorder, interval histograms, log writer, viewers
  - DoubleHistogram h = new DoubleHistogram(3);

AZUL SYSTEMS

HdrHistogram: How did it come to be?

- It started as an inner class in a "JitterMeter"
  - Needed a way to deal with logging and reporting latencies across a wide range. ~100 LOC.
- I kept copying it out to other tools
- So I made a class out of it
  - Then I showed it to a friend
- Next thing I know:
  - I'm running a Yak-shaving co-op on github
  - The first Yak hair I produced was Java-colored Himalayan but others have since added other colors and breeds

hdrhistogram.org

HdrHistogram

A High Dynamic Range (HDR) Histogram

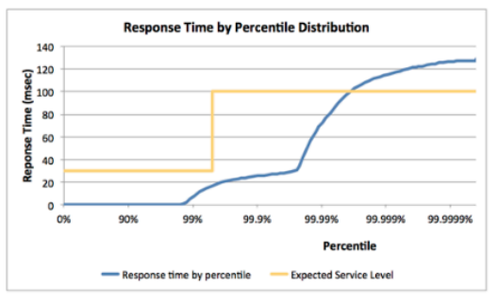
View the JavaDoc
hdrhistogram.github.com/HdrHistogram/JavaDoc
Find on Maven Central
org.hdrhistogram

View implementations on GitHub:

Java
HdrHistogram/HdrHistogram
JavaScript
HdrHistogram/HdrHistogramJS
C
HdrHistogram/HdrHistogram_c
C#/.NET
HdrHistogram/HdrHistogram.NET
Python
HdrHistogram/HdrHistogram_py
Erlang
HdrHistogram/hdr_histogram_erl
Rust
HdrHistogram/HdrHistogram_rust
Go
codahale/hdrhistogram

Plot histogram file(s)

An example plot of an HdrHistogram based full percentile spectrum plot:



HdrHistogram: A High Dynamic Range Histogram.

A Histogram that supports recording and analyzing sampled data value counts across a configurable integer value range with configurable value precision within the range. Value precision is expressed as the number of significant digits in the value recording, and provides control over value quantization behavior across the value range and the subsequent value resolution at any given level.

For example, a Histogram could be configured to track the counts of observed integer values between 0 and 3,600,000,000 while maintaining a value precision of 3 significant digits across that range. Value quantization within the range will thus be no larger than 1/1,000th (or 0.1%) of any value. This example Histogram could be used to track and analyze the counts of observed response times ranging between 1 microsecond and 1 hour in magnitude, while maintaining a value resolution of 1 microsecond up to 1 millisecond, a resolution of 1 millisecond (or better) up to one second, and a resolution of 1 second (or better) up to 1,000 seconds. At it's maximum tracked value (1 hour), it would still maintain a resolution of 3.6 seconds (or better).

HDR Histogram is designed for recoding histograms of value measurements in latency and performance sensitive applications. Measurements show value recording times as low as 3-6 nanoseconds on modern (circa 2014) Intel CPUs. The HDR Histogram maintains a fixed cost in both space and time. A Histogram's memory footprint is constant, with no allocation operations involved in recording data values or in iterating through them. The memory footprint is fixed regardless of the number of data value samples recorded, and depends solely on the dynamic range and precision chosen. The amount of work involved in recording a sample is constant, and directly computes storage index locations such that no iteration or searching is ever involved in recording data values.

Authors, Contributors, and License

HdrHistogram was originally authored by Gil Tene (@giltene) (Java) and placed in the public domain, as explained at http://creativecommons.org/publicdomain/zero/1.0/, with ports by Mike Barker (@mikeb01) (C), Matt Warren (@mattwarren) (C#), Darach Ennis (@darach) (Erlang), Alec Hothan (@ahothan) (Python), Coda Hale (@codahale) (Go), Jon Gjengset (@jonhoo) (Rust), and Alexandre Victoor (@alexvictoor) (JavaScript)

Support or Contact

Don't call me, I won't call you.

AZUL SYSTEMS

Search or jump to... / Pull requests Issues Marketplace Explore

HdrHistogram / HdrHistogram

Unwatch 115 | Unstar 1,533 | Fork 176

Code | Issues 9 | Pull requests 4 | Projects 0 | Wiki | Insights | Settings

A High Dynamic Range (HDR) Histogram http://hdrhistogram.github.com/HdrHis... Edit

Manage topics

656 commits | 4 branches | 25 releases | 1 environment | 30 contributors | View license

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

giltene Comment touchups — Latest commit e692d03 18 days ago

GoogleChartsExample	small cleanups to pacify code quality tools	4 months ago
HdrHistogram-benchmarks	Moved to jmh 1.6, which works with both Java 6 and Java 9	a year ago
gnuplotExample	Modified make_percentile_plot to work when /bin/sh does not behave as...	4 years ago
lib/test	Added junit lib	6 years ago
src	Comment touchups	18 days ago
.gitignore	Update .NET to VS2015 format	3 years ago
.travis.yml	Prune OpenJDK 9 from CI list. If 6 is not on there, 9 certainly shoul...	20 days ago
COPYING.txt	JavaDoc and license cosmetics	7 years ago
HistogramLogProcessor	stray JHICCUP replaced with HDRHISTOGRAM	3 years ago
LICENSE.txt	Updates copyright years	2 years ago

First commit: Sep. 11, 2012

AZUL SYSTEMS

README.md

HdrHistogram

HdrHistogram: A High Dynamic Range (HDR) Histogram

This respository currently includes a Java implementation of HdrHistogram. C, C#/.NET, Python, Javascript, Rust, Erlang, and Go ports can be found in other respositories. All of which share common concepts and data representation capabilities. Look at repositories under the HdrHistogram organization for various implementations and useful tools.

Note: The below is an excerpt from a Histogram JavaDoc. While much of it generally applies to other language implementations as well, some details may vary by implementation (e.g. iteration and synchronization), so you should consult the documentation or header information of specific API library you intended to use.

HdrHistogram supports the recording and analyzing of sampled data value counts across a configurable integer value range with configurable value precision within the range. Value precision is expressed as the number of significant digits in the value recording, and provides control over value quantization behavior across the value range and the subsequent value resolution at any given level.

For example, a Histogram could be configured to track the counts of observed integer values between 0 and 3,600,000,000 while maintaining a value precision of 3 significant digits across that range. Value quantization within the range will thus be no larger than 1/1,000th (or 0.1%) of any value. This example Histogram could be used to track and analyze the counts of observed response times ranging between 1 microsecond and 1 hour in magnitude, while maintaining a value resolution of 1 microsecond up to 1 millisecond, a resolution of 1 millisecond (or better) up to one second, and a resolution of 1 second (or better) up to 1,000

AZUL SYSTEMS

Home » org.hdrhistogram » HdrHistogram

HdrHistogram

HdrHistogram supports the recording and analyzing sampled data value counts across a configurable integer value range with configurable value precision within the range. Value precision is expressed as the number of significant digits in the value recording, and provides control over value quantization behavior across the value range and the subsequent value resolution at any given level.

License	BSD 2-clause CC0 1.0
Used By	**207 artifacts**

Central (24) | Redhat GA (2) | Redhat EA (1)

	Version	Repository	Usages	Date
2.1.x	2.1.11	Central	7	Jan, 2019
	2.1.10	Central	60	Sep, 2017
	2.1.9	Central	79	May, 2016
	2.1.8	Central	45	Nov, 2015
	2.1.7	Central	8	Sep, 2015
	2.1.6	Central	12	Jul, 2015
	2.1.5	Central	0	Jul, 2015
	2.1.4	Central	29	Jan, 2015
	2.1.3	Central	3	Dec, 2014
	2.1.2	Central	6	Dec, 2014
	2.1.1	Central	0	Dec, 2014
	2.1.0	Central	12	Dec, 2014
2.0.x	2.0.3	Central	3	Nov, 2014
	2.0.2	Central	0	Nov, 2014
	2.0.1	Central	1	Nov, 2014
1.2.x	1.2.1	Central	9	May, 2014
1.1.x	1.1.6	Central	0	May, 2014
	1.1.5	Central	1	May, 2014
	1.1.4	Central	1	Apr, 2014
	1.1.3	Central	0	Apr, 2014
1.0.x	1.0.9	Central	1	Jan, 2014
	1.0.8	Central	2	Nov, 2013
	1.0.7	Central	0	Nov, 2013
	1.0.6	Central	0	Nov, 2013

AZUL SYSTEMS

Artifacts using HdrHistogram (207)

Sort: **popular** | newest

1. **Log4j Core** — **3,813** usages
org.apache.logging.log4j » log4j-core — Apache
Log4j Core
Last Release on Jul 30, 2018

2. **Akka Testkit** — **1,243** usages
com.typesafe.akka » akka-testkit — Apache
akka-testkit
Last Release on Jan 29, 2019

3. **Server** — **875** usages
org.elasticsearch » elasticsearch — Apache
Elasticsearch subproject :server
Last Release on Dec 20, 2018

4. **DataStax Java Driver For Apache Cassandra Core** — **555** usages
com.datastax.cassandra » cassandra-driver-core — Apache
A driver for Apache Cassandra 1.2+ that works exclusively with the Cassandra Query Language version 3 (CQL3) and Cassandra's binary protocol.
Last Release on Aug 29, 2018

5. **DataStax Java Driver For Apache Cassandra Core** — **555** usages
com.datastax.cassandra » cassandra-driver-core — Apache
A driver for Apache Cassandra 1.2+ that works exclusively with the Cassandra Query Language version 3 (CQL3) and Cassandra's binary protocol.
Last Release on Aug 29, 2018

6. **DataStax Java Driver For Apache Cassandra Core** — **555** usages
com.datastax.cassandra » cassandra-driver-core — Apache
A driver for Apache Cassandra 1.2+ that works exclusively with the Cassandra Query Language version 3 (CQL3) and Cassandra's binary protocol.
Last Release on Aug 29, 2018

7. **Hazelcast** — **366** usages
com.hazelcast » hazelcast — Apache
Core Hazelcast Module
Last Release on Dec 18, 2018

8. **Hazelcast** — **366** usages
com.hazelcast » hazelcast — Apache
Core Hazelcast Module
Last Release on Dec 18, 2018

9. **Hystrix Core** — **227** usages
com.netflix.hystrix » hystrix-core — Apache
hystrix-core

AZUL SYSTEMS

Libraries

do you want to build one?

AZUL SYSTEMS

Libraries

do you want to build one?

- Well…
- What do you like to do with your weekends?
  - If you don't get paid to do this…
- If it could be part of your job
  - Do you want it to be part of your job?
- If so, prepare for some realities…
- For example, are you willing to code to Java 7?

AZUL SYSTEMS

Libraries

Is it a good idea?

AZUL SYSTEMS

Libraries

Is it a good idea?

- IDK
- What makes it "needed"
- Is it just that you don't like the 7 other libraries that do the same thing you are about to build?
- Or is it something that is missing
- Best positive-indication symptom IMO:
  - Other people (choose to) copy your code
- Still, will it make it?

AZUL SYSTEMS

Libraries

Will it "make it"?

AZUL SYSTEMS

Libraries

Will it "make it"?

- IMO, most libraries die young
  - For every star and fork on github, there are probably 2 dead libraries that never got adopted
  - And that's OK. It is as it should be.
- Don't get discouraged
- Don't get your hopes up
- Do a good job
  - Think of what adopters will expect and need
- Think of rules to follow, and start early

Libraries

rules that only library writer have to follow

AZUL SYSTEMS

Libraries

rules that only library writer have to follow

- Examples only…
- API compatibility
  - Including binary compatibility
    - e.g. a default interface method can be "interesting"
- Plan to start by coding to the "the mainstream" production language versions of ~3 years ago
- If someone starts using your library today
  - they will hopefully keep using it in 3, 5, 8 years
  - So expect coding to Java versions that are ~7-10 years old

AZUL SYSTEMS

Libraries

Can be hard to get rid of

AZUL SYSTEMS

giltene / wrk2

Unwatch 5 | Star 1,726 | Fork 146

Code | Issues 32 | Pull requests 7 | Projects 0 | Wiki | Insights | Settings

A constant throughput, correct latency recording variant of wrk

Edit

Manage topics

110 commits | 1 branch | 0 releases | 16 contributors | Apache-2.0

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

giltene Merge pull request #18 from alex-koturanov/master ... Latest commit e0109df on Mar 3, 2018

CoordinatedOmission	Added image depicting difference between CO and clean wi...	4 years ago
deps/luajit	import LuaJIT 2.0.3	5 years ago
scripts	Pulling in more changes from wrk fixing host header	3 years ago
src	Merge pull request #18 from alex-koturanov/master	11 months ago
.gitignore	Updates .gitignore to work with CLion	4 years ago
LICENSE	first public release of wrk	7 years ago
Makefile	Merge pull request #14 from blu2lz/master	11 months ago
NOTICE	Touched up LICENSE	4 years ago
README.md	Merge pull request #35 from icflournoy/readme_typo_fixes	11 months ago
SCRIPTING	Pulling in more changes from wrk fixing host header	3 years ago

README.md

wrk2

gitter join chat

a HTTP benchmarking tool based mostly on wrk

wrk2 is wrk modifed to produce a constant throughput load, and accurate latency details to the high 9s (i.e. can produce accurate 99.9999%'ile when run long enough). In addition to wrk's arguments, wrk2 takes a throughput argument (in total requests per second) via either the --rate or -R p

wrk2 is currently in experimental/development mode, and may well be merged into wrk in the future if others see fit to adopt it's changes.

AZUL SYSTEMS

Home » org.latencyutils » LatencyUtils

LatencyUtils

LatencyUtils is a package that provides latency recording and reporting utilities.

License	CC0 1.0
Used By	**25 artifacts**

Central (9)

	Version	Repository	Usages	Date
2.0.x	2.0.3	Central	11	Dec, 2015
	2.0.2	Central	14	Dec, 2014
	2.0.1	Central	0	Nov, 2014
1.1.x	1.1.1	Central	0	Nov, 2014
1.0.x	1.0.5	Central	2	May, 2014
	1.0.4	Central	0	May, 2014
	1.0.3	Central	2	Dec, 2013
	1.0.2	Central	0	Nov, 2013
	1.0.1	Central	0	Nov, 2013

AZUL SYSTEMS

Home » org.latencyutils » LatencyUtils » Usages

Artifacts using LatencyUtils (25)

Sort: **popular** | newest

1. **Akka Testkit** — **1,243** usages — Apache
com.typesafe.akka » akka-testkit
akka-testkit
Last Release on Jan 29, 2019

2. **Micrometer Core** — **135** usages — Apache
io.micrometer » micrometer-core
Application monitoring instrumentation facade
Last Release on Jan 11, 2019

3. **Quasar Core** — **65** usages — LGPL | EPL
co.paralleluniverse » quasar-core
Fibers, Channels and Actors for the JVM
Last Release on Nov 2, 2018

4. **Quasar Core** — **65** usages — LGPL | EPL
co.paralleluniverse » quasar-core
Fibers, Channels and Actors for the JVM
Last Release on Nov 2, 2018

5. **Lettuce** — **60** usages — Apache
io.lettuce » lettuce-core
Advanced and thread-safe Java Redis client for synchronous, asynchronous, and reactive usage. Supports Cluster, Sentinel, Pipelining, Auto-Reconnect, Codecs and much more.
Last Release on Nov 26, 2018

6. **Infinispan Query** — **47** usages — Apache
org.infinispan » infinispan-query
Infinispan Query module
Last Release on Jan 28, 2019

7. **Infinispan Query** — **47** usages — Apache
org.infinispan » infinispan-query
Infinispan Query module
Last Release on Jan 28, 2019

8. **Micrometer Core** — **135** usages — Apache
io.micrometer » micrometer-core
Application monitoring instrumentation facade
Last Release on Jan 11, 2019

AZUL SYSTEMS

Libraries

Can be hard to get rid of

- Issues pop up and people will beg you for stuff
- Does it work on Java 11?
- What about 12?
- Can you keep it working for Java 7 and 11 at the same time?
- What about 8 and 23?
- Tricks of the trade

AZUL SYSTEMS

HdrHistogram
(as an anecdotal example)

- Ticket opened on Java implementation: Broke on Java 9
- Fixing it broke Java 7….
- … … … …
- Eventually built helper class to bridge gaps.
- Would I have done this if HdrHistogram had any dependencies?
  - Probably not…
- But lets try to save others some time…

AZUL SYSTEMS

Problem statement: Library portability

- Library needs a single jar that works on Java 7/8/9/10...
- Makes use of APIs that do not exist (in the same form) in all JDK versions
- Multi-version jar (alone) solves nothing
  - Split-up sources are not maintainable
- A "good" fix will encapsulate problem areas
  - Solve in "ugly" but internal, well-encapsulated way
  - E.g. reflection, method handles, multi-version-jars, ...

org.HdrHistiogram.Base64Helper

```
static String printBase64Binary(byte [] binaryArray) {
```

```
static byte[] parseBase64Binary(String base64input) {
```


AZUL SYSTEMS

org.HdrHistiogram.Base64Helper

```
/**
 * Base64Helper exists to bridge inconsistencies in Java SE support of Base64 encoding and decoding.
 * Earlier Java SE platforms (up to and including Java SE 8) supported base64 encode/decode via the
 * javax.xml.bind.DatatypeConverter class, which was deprecated and eventually removed in Java SE 9.
 * Later Java SE platforms (Java SE 8 and later) support base64 encode/decode via the
 * java.util.Base64 class (first introduced in Java SE 8, and not available on e.g. Java SE 6 or 7).
 *
 * This makes it "hard" to write a single piece of source code that deals with base64 encodings and
 * will compile and run on e.g. Java SE 7 AND Java SE 9. And such common source is a common need for
 * libraries. This class is intended to encapsulate this "hard"-ness and hide the ugly pretzle-twising
 * needed under the covers.
 *
 * Base64Helper provides a common API that works across Java SE 6..9 (and beyond hopefully), and
 * uses late binding (Reflection) internally to avoid javac-compile-time dependencies on a specific
 * Java SE version (e.g. beyond 7 or before 9).
 *
 */
public class Base64Helper {

    static String printBase64Binary(byte [] binaryArray) {

    static byte[] parseBase64Binary(String base64input) {
```


AZUL SYSTEMS

org.HdrHistiogram.Base64Helper

```
/**
 * Converts an array of bytes into a Base64 string.
 *
 * @param binaryArray A binary encoded input array
 * @return a String containing the Base64 encoded equivalent of the binary input
 */
static String printBase64Binary(byte [] binaryArray) {
    try {
        return (String) encodeMethod.invoke(encoderObj, binaryArray);
    } catch (Throwable e) {
        throw new UnsupportedOperationException("Failed to use platform's base64 encode method");
    }
}

/**
 * Converts a Base64 encoded String to a byte array
 *
 * @param base64input A base64-encoded input String
 * @return a byte array containing the binary representation equivalent of the Base64 encoded input
 */
static byte[] parseBase64Binary(String base64input) {
    try {
        return (byte []) decodeMethod.invoke(decoderObj, base64input);
    } catch (Throwable e) {
        throw new UnsupportedOperationException("Failed to use platform's base64 decode method");
    }
}
```


AZUL SYSTEMS

org.HdrHistiogram.Base64Helper: the "ugly"

```
private static Method decodeMethod;
private static Method encodeMethod;

// encoderObj and decoderObj are used in non-static method forms, and
// irrelevant for static method forms:
private static Object decoderObj;
private static Object encoderObj;

static {
    try {
        Class<?> javaUtilBase64Class = Class.forName("java.util.Base64");

        Method getDecoderMethod =  javaUtilBase64Class.getMethod("getDecoder");
        decoderObj = getDecoderMethod.invoke(null);
        decodeMethod = decoderObj.getClass().getMethod("decode", String.class);

        Method getEncoderMethod =  javaUtilBase64Class.getMethod("getEncoder");
        encoderObj = getEncoderMethod.invoke(null);
        encodeMethod = encoderObj.getClass().getMethod("encodeToString", byte[].class);
    } catch (Throwable e) {
        decodeMethod = null;
        encodeMethod = null;
    }

    if (encodeMethod == null) {
        decoderObj = null;
        encoderObj = null;
        try {
            Class<?> javaxXmlBindDatatypeConverterClass = Class.forName("javax.xml.bind.DatatypeConverter");
            decodeMethod = javaxXmlBindDatatypeConverterClass.getMethod("parseBase64Binary", String.class);
            encodeMethod = javaxXmlBindDatatypeConverterClass.getMethod("printBase64Binary", byte[].class);
        } catch (Throwable e) {
            decodeMethod = null;
            encodeMethod = null;
        }
    }
}
```


AZUL SYSTEMS

I can't be the only one, right?

And these are probably not the only two methods…

AZUL SYSTEMS

Problem statement: pre-coding for new features

- A new capability is anticipated in a coming JDK
- We want libraries and projects to pre-code with new feature in mind
- But they need to run on existing JDKs
- Solution will tend to encapsulate new functionality
  - E.g. method handles, or pure-java semantic equivalent

Thread.onSpinWait()

```
/**
 * Indicates that the caller is momentarily unable to progress, until the
 * occurrence of one or more actions on the part of other activities. By
 * invoking this method within each iteration of a spin-wait loop construct,
 * the calling thread indicates to the runtime that it is busy-waiting.
 * The runtime may take action to improve the performance of invoking
 * spin-wait loop constructions.
 *
 * @apiNote
 * As an example consider a method in a class that spins in a loop until
 * some flag is set outside of that method. A call to the {@code onSpinWait}
 * method should be placed inside the spin loop.
 * <pre>{@code
 *     class EventHandler {
 *         volatile boolean eventNotificationNotReceived;
 *         void waitForEventAndHandleIt() {
 *             while ( eventNotificationNotReceived ) {
 *                 java.lang.Thread.onSpinWait();
 *             }
 *             readAndProcessEvent();
 *         }
 *
 *         void readAndProcessEvent() {
 *             // Read event from some source and process it
 *              . . .
 *         }
 *     }
 * }</pre>
 * <p>
 * The code above would remain correct even if the {@code onSpinWait}
 * method was not called at all. However on some architectures the Java
 * Virtual Machine may issue the processor instructions to address such
 * code patterns in a more beneficial way.
 *
 * @since 9
 */
@HotSpotIntrinsicCandidate
public static void onSpinWait() {}
```

Thread.onSpinWait()

```
public static void onSpinWait() {}
```

org.agrona.hints.ThreadHints

AZUL SYSTEMS

org.agrona.hints.ThreadHints

```java
public final class ThreadHints
{
    private static final MethodHandle ON_SPIN_WAIT_METHOD_HANDLE;

    static
    {
        final MethodHandles.Lookup lookup = MethodHandles.lookup();

        MethodHandle methodHandle = null;
        try
        {
            methodHandle = lookup.findStatic(Thread.class, "onSpinWait", methodType(void.class));
        }
        catch (final Exception ignore)
        {
        }

        ON_SPIN_WAIT_METHOD_HANDLE = methodHandle;
    }

    private ThreadHints()
    {
    }

    /**
     * Indicates that the caller is momentarily unable to progress, until the
     * occurrence of one or more actions on the part of other activities.  By
     * invoking this method within each iteration of a spin-wait loop construct,
     * the calling thread indicates to the runtime that it is busy-waiting. The runtime
     * may take action to improve the performance of invoking spin-wait loop constructions.
     */
    public static void onSpinWait()
    {
        // Call java.lang.Thread.onSpinWait() on Java SE versions that support it. Do nothing otherwise.
        // This should optimize away to either nothing or to an inlining of java.lang.Thread.onSpinWait()
        if (null != ON_SPIN_WAIT_METHOD_HANDLE)
        {
            try
            {
                ON_SPIN_WAIT_METHOD_HANDLE.invokeExact();
            }
            catch (final Throwable ignore)
            {
            }
        }
    }
}
```

org.agrona.hints.ThreadHints

agrona/src/main/java/org/agrona/concurrent/BusySpinIdleStrategy.java

Showing the top match Last indexed on Mar 9

```
36                 return;
37             }
38
39             idle();
40         }
41
42         public void idle()
43         {
44             ThreadHints.onSpinWait();
```

agrona/src/main/java/org/agrona/concurrent/ControllableIdleStrategy.java

Showing the top match Last indexed on Mar 9

```
68                 case BUSY_SPIN:
69                     ThreadHints.onSpinWait();
70                     break;
71
72                 case YIELD:
73                     Thread.yield();
```

agrona/src/main/java/org/agrona/concurrent/BackoffIdleStrategy.java

Showing the top match Last indexed on Mar 9

```
103                    break;
104
105                case SPINNING:
106                    ThreadHints.onSpinWait();
107                    if (++spins > maxSpins)
```


AZUL SYSTEMS

My next Idea for a Library

- "Safe"
- As in: org.javaportability.Safe
- A library that does "Unsafe" things safely on Java 11+
- But uses sun.misc.Unsafe under the hood for 8-

AZUL SYSTEMS