Easy High Constancy

Circuit breaking for fault tolerance

 Dallas Clement  Sergey Nepomnyachiy

“Man is mortal.
The worst of it — he's unexpectedly mortal.”

M. A. Bulgakov

Motivation

We all make calls to:

- Algorithms *with awful worst case running time*
- Databases *on a failing hardware*
- Remote services *across unreliable networks*

Motivation

Consequently, the above may result in various faults:

- Timeouts
- Exceptions
- Invalid values

Goals

We need a library for instrumenting the calls, such that the faults are:

- Acknowledged and recorded
- Alleviated by validation and normalization
- Avoided by "circuit breaking" — skipping the invocation entirely

Prior art

- Netflix Hystrix for Java
- Pycopine — Python port

Our Contribution

Eazy High Constancy

- C++ library with no dependencies
- C++03 compliant
- Open source, MIT license

github.com/ezhic/ezhic

Usage example

Existing call:

```
remote.call(arg1, arg2);
```

Usage example

Instrumented call with a label:

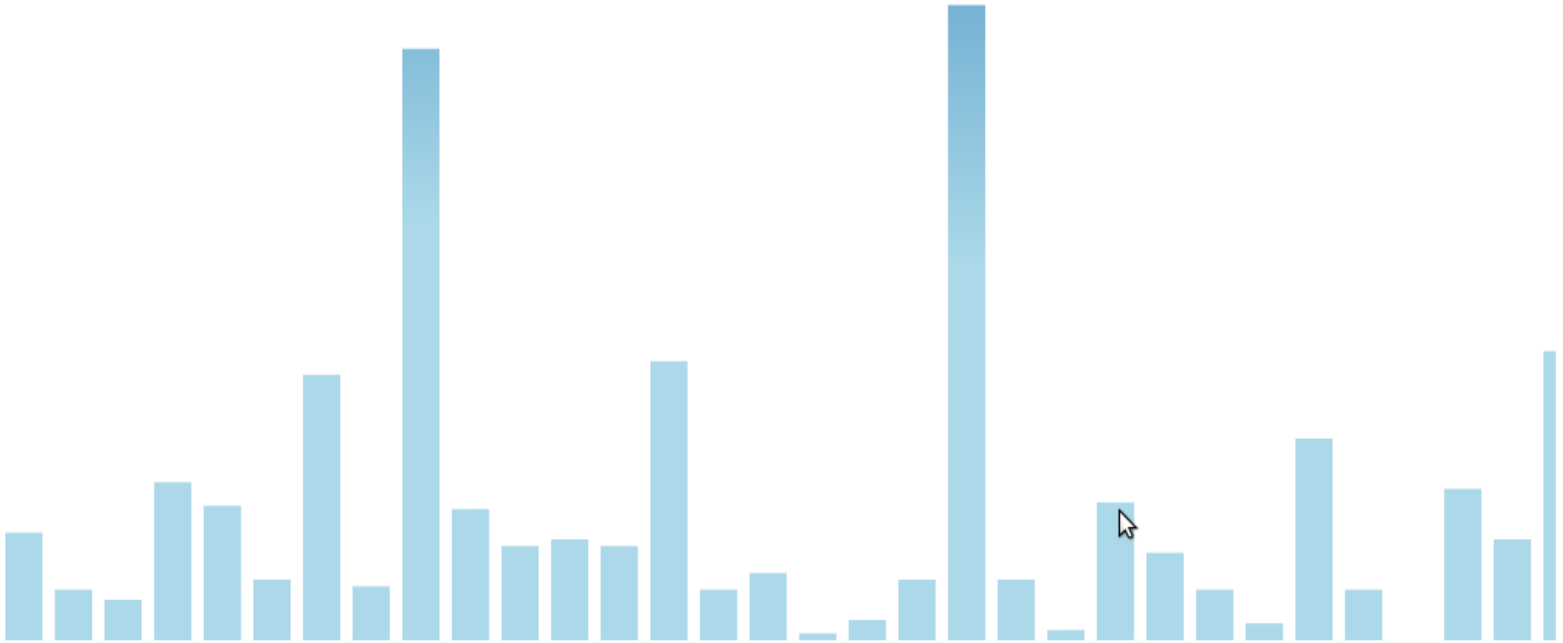
```
EZ_START("label-for-remote1");

remote.call(arg1, arg2);

EZ_END();
```

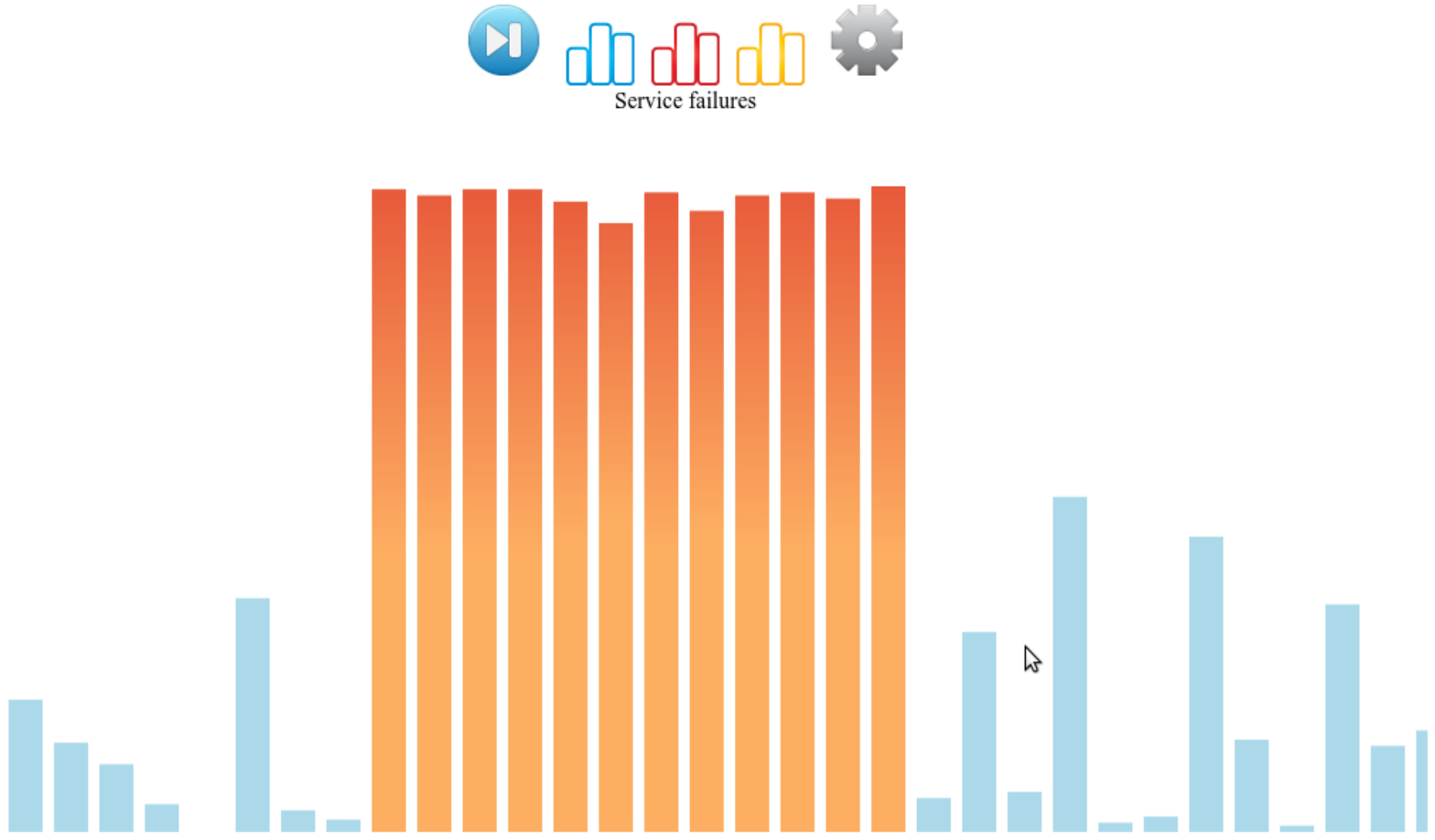
Usage example

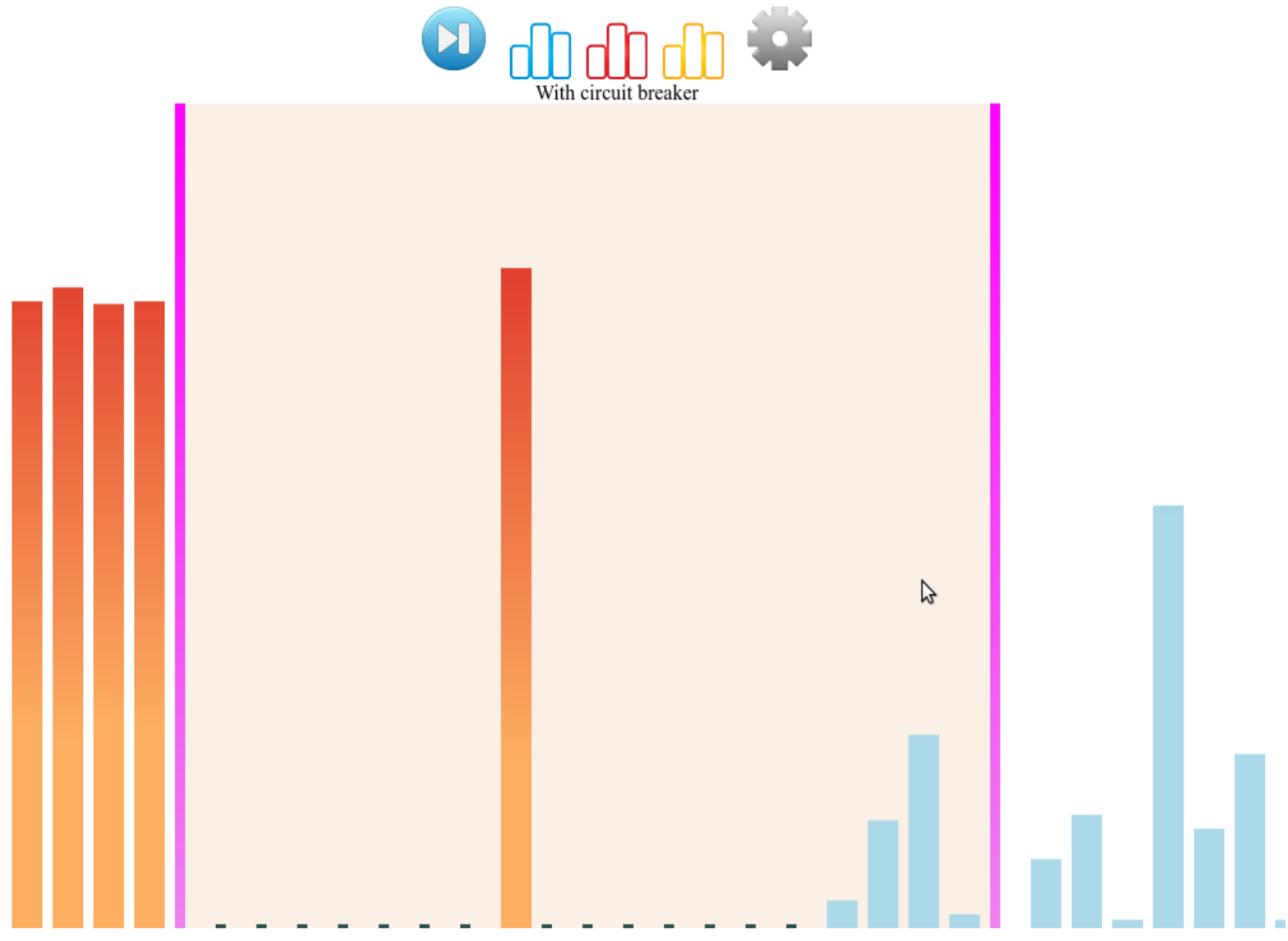
Associating config. bundle with a label:

```
Bundle bundle;
ezreg::writeBundle("label-for-remote1", bundle);

...

BRunner runner("label-for-remote1");
runner.run_m(remote, &Remote::call, arg1, arg2);
```

Uneventful




Service failures


With circuit breaker

Future work

- C++11 extension
- Asynchronous scheduler

Easy High Constancy

Circuit breaking for fault tolerance

 Dallas Clement  Sergey Nepomnyachiy

“Man is mortal.
The worst of it — he's unexpectedly mortal.”

M. A. Bulgakov

Motivation

We all make calls to:

- Algorithms *with awful worst case running time*
- Databases *on a failing hardware*
- Remote services *across unreliable networks*

Motivation

Consequently, the above may result in various faults:

- Timeouts
- Exceptions
- Invalid values

Goals

We need a library for instrumenting the calls, such that the faults are:

- Acknowledged and recorded
- Alleviated by validation and normalization
- Avoided by "circuit breaking" — skipping the invocation entirely

Prior art

- Netflix Hystrix for Java
- Pycopine — Python port

Our Contribution

EzHiC

Eazy **Hi**gh **C**onstancy

- C++ library with no dependencies
- C++03 compliant
- Open source, MIT license

github.com/ezhic/ezhic

Usage example

Existing call:

```
remote.call(arg1, arg2);
```

Usage example

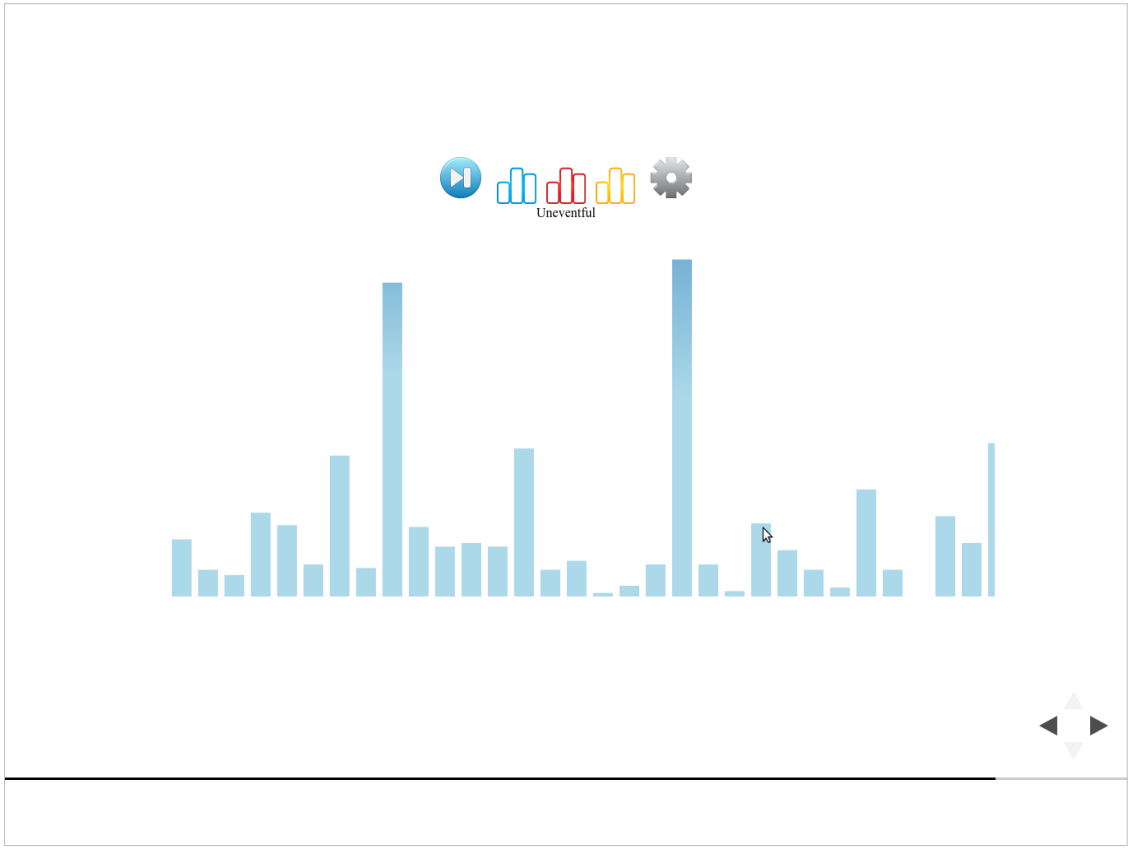
Instrumented call with a label:

```
EZ_START("label-for-remote1");
remote.call(arg1, arg2);
EZ_END();
```

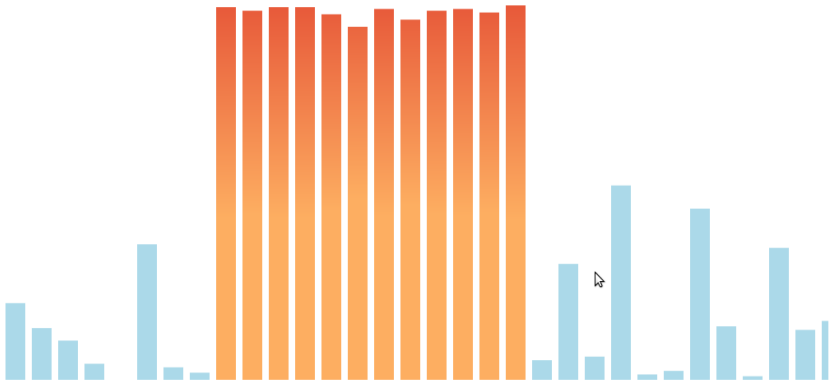
Usage example

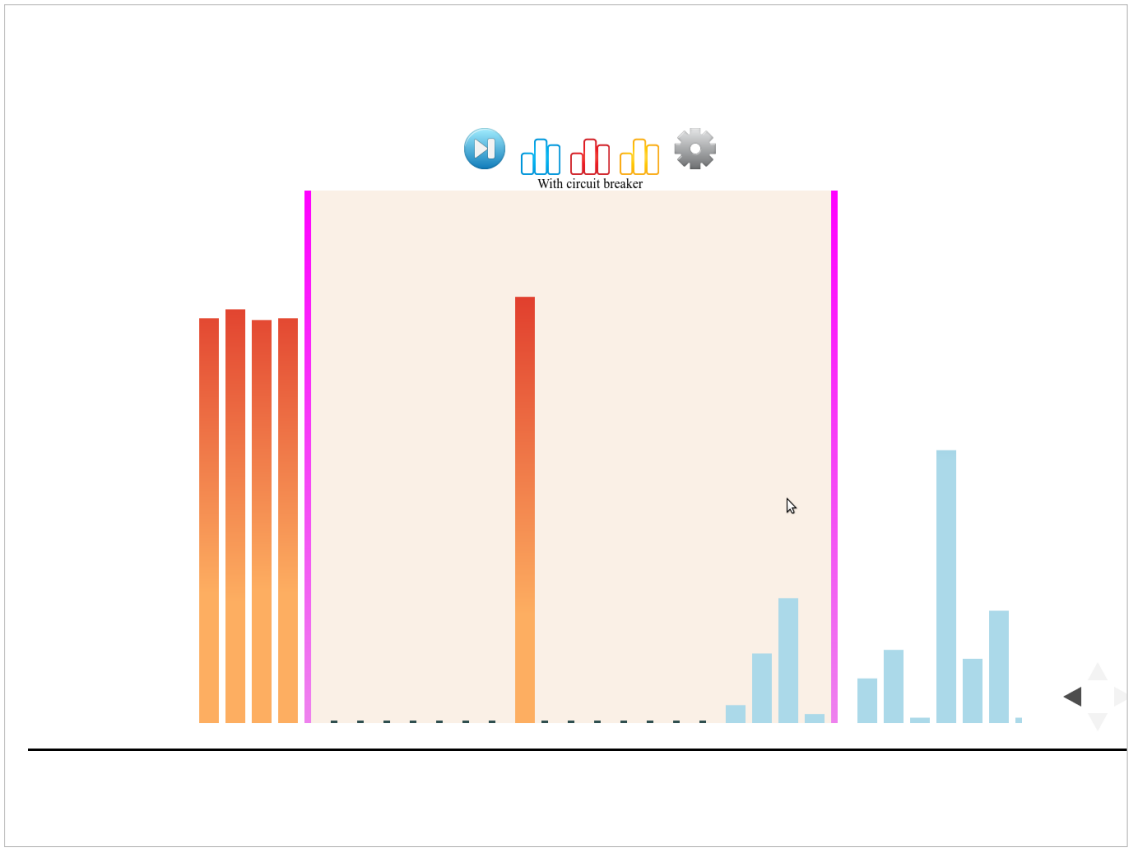
Associating config. bundle with a label:

```
Bundle bundle;
ezreg::writeBundle("label-for-remote1", bundle);

...

BRunner runner("label-for-remote1");
runner.run_m(remote, &Remote::call, arg1, arg2);
```


Uneventful

Service failures




With circuit breaker

Future work

- C++11 extension
- Asynchronous scheduler