BlkKin: A Low-overhead tracing infrastructure for software-defined storage systems

Marios-Evangelos Kogias

National Technical University of Athens
School of Electrical and Computer Engineering

October 16, 2014

Outline

Thesis Background

synnefo

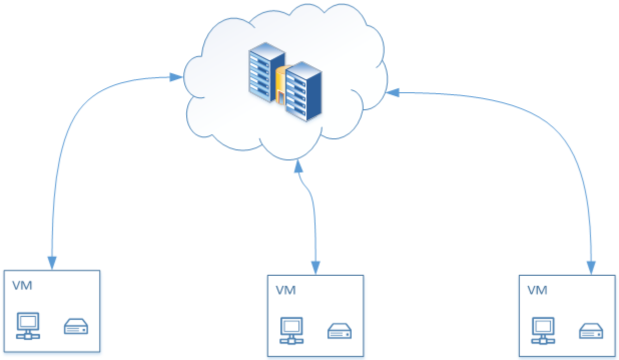
Open source, production-ready, cloud software.
Designed since 2010 by GRNET.

okeanos

- IaaS service
- Targeted at the Greek Academic and Research Community
- Designed by GRNET
- In production since 2011

Outline

VM Volume storage

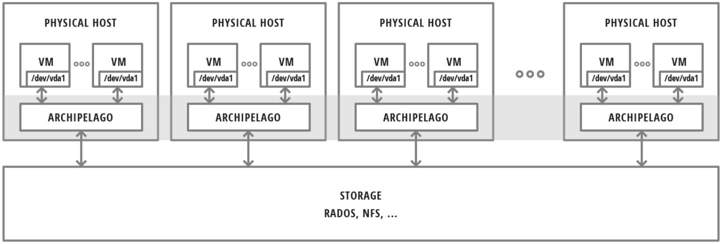


Archipelago I

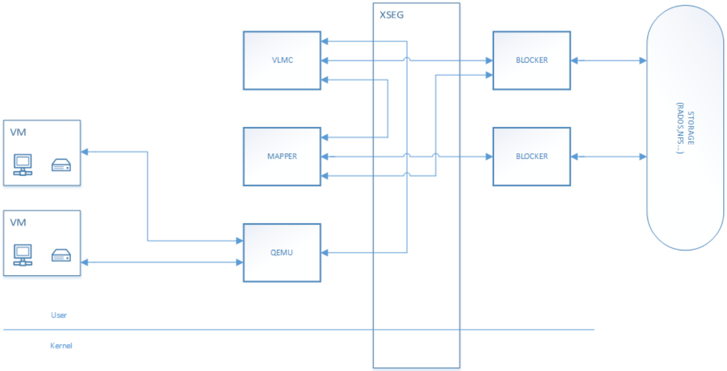
A thin distributed storage layer aiming to:

- Decouple storage logic from the actual data store
- Provide logic for thin cloning and snapshotting
- Provide logic for deduplication
- Provide different endpoint drivers to access Volumes and Files
- Provide backend drivers for different storage technologies

Archipelago II



Archipelago III



RADOS

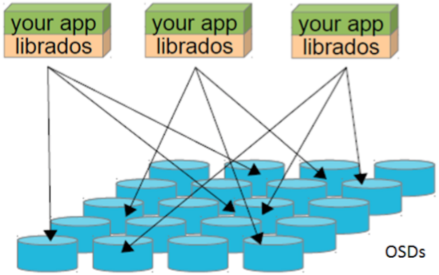
is the storage component of Ceph

Ceph is a distributed object store and filesystem.

RADOS basic characteristics are:

- Replication
- Fault tolerance
- Self-management
- Scalability

Storage Abstraction



The Problem

- Complex service-oriented architectures
- Difficult to debug
- Difficult to monitor
- Non-deterministic execution
- Context-dependent faults

Solution

Distributed end-to-end tracing

&

Central data collection

BlkKin

A distributed tracing infrastructure to track the I/O request from QEMU until RADOS

BlkKin's main characteristics:

- low-overhead tracing
- live-tracing
- End-to-end tracing of causal relationships
- User interface

Outline

Main Challenges

- Meaningful and easily correlated tracing data
- Low overhead tracing backend

Schools of thought

black-box schemes
: They assume there is no additional information other than the message record described above and use statistical regression techniques to infer that association.

annotation-based schemes
: They rely on applications or middleware to explicitly tag every record with a global identifier that links these message records back to the originating request.

The Dapper System

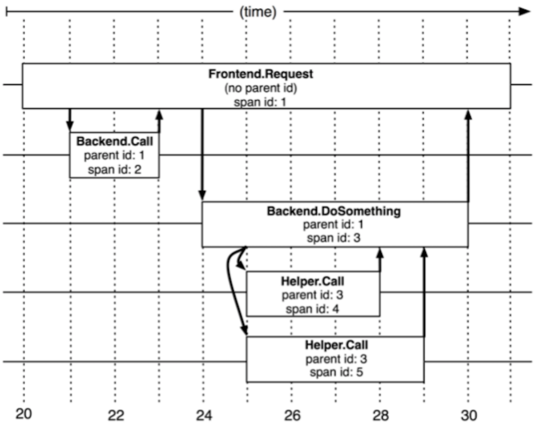
- Large-scale distributed systems tracing infrastructure created by Google[1]
- Annotation-based tracing scheme
- Common libraries instrumentation
  - RPC System
  - Control Flow
- BigTable backend
- Closed-source

[1] Benjamin H. Sigelman et al. *Dapper, a Large-Scale Distributed Systems Tracing Infrastructure*. Tech. rep. Google, Inc., 2010. URL: http://research.google.com/archive/papers/dapper-2010-1.pdf.

Dapper tracing concepts

annotation	The actual information being logged. Either *timestamp* or *key-value*
span	The basic unit of the process tree. Can represent a subsystem or a function call. To depict causal relationship each span has a parent span or is a *root* span.
trace	A different trace id is used to group data related to the same initial request

Causal realtionships



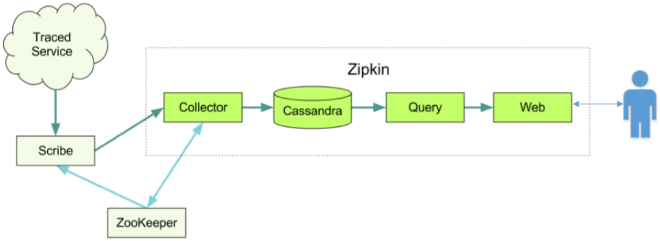
Zipkin

An open-source Scala implementation of the Dapper paper by Twitter

Zipkin services:

- Data collector
- Database service
- Web UI

Zipkin Architecture



Scribe

Scribe is a scalable and reliable logging server created by Facebook

- Written in C++
- Directed graph architecture
- Batch messaging
- HDFS support
- Based on Apache Thrift

Thrift

A software framework for scalable cross-language services development.

Includes a code generation engine to create RPC services across programming languages based on a Thrift file

Sample target languages: C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, OCaml

Zipkin sum up

Zipkin	is a full-stack tracing system using
Scribe	as its logging server using
Thrift	as its transport protocol

Tracing

"Tracing is a specialized use of logging to record information about a program's execution"

Wikipedia

Tracing characteristics:

- Tracing can be low level (eg. kernel tracing, access to performance counters)
- Tracing has mostly debugging purposes and performance tuning
- Tracing may produce notoriously bulky output

Tracing Systems

DTrace

Released by Sun Microsystems in 2005

SystemTap

Released by Red Hat in 2005

Advantages:

- Dynamic Instrumentation
- User and kernel tracing

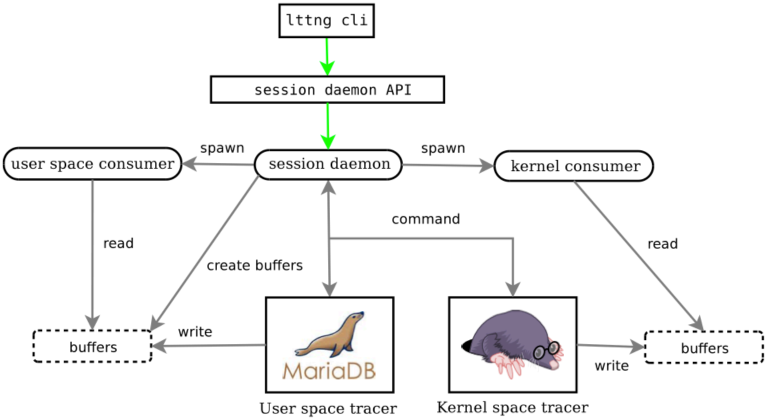
Disadvantages:

- User tracing is based or system calls or breakpoints
- Significant performance overhead
- Inappropriate for live tracing

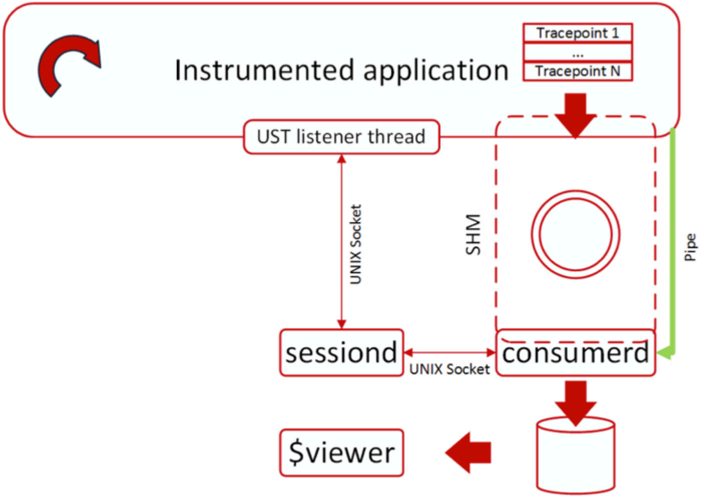
Linux Trace Toolkit - next generation

- successor of Linux Trace Toolkit
- Mathieu Desnoyers' PhD dissertation in Ecole Polytechnique de Montreal
- maintained by EfficiOS Inc and the DORSAL lab in Ecole Polytechnique de Montreal.
- Unified user and kernel tracing
- Low overhead tracing based on Tracepoints
- Static instrumentation
- Live tracing

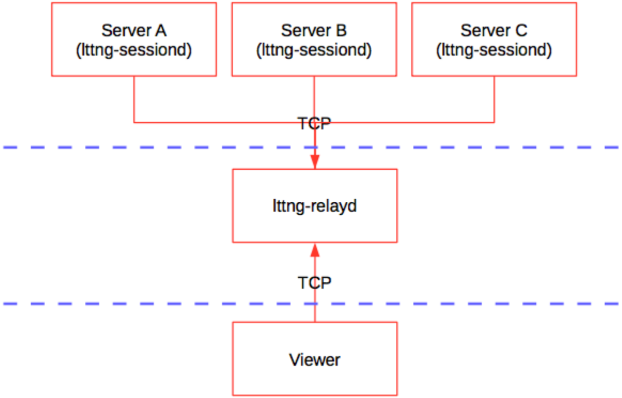
LTTng Unified Tracing



UST architecture



LTTng live-tracing



CTF

Common Trace Format:

- Aimed to cover tracing needs from versatile communities
- Collaboration between the Multicore association and the Linux Community
- Based on Trace Stream Description Language
- Separation between data and metadata
- Separation between event and event-context
- Variety of data types and type inheritance
- Live tracing

Babeltrace

- Trace reader/writer
- Trace converter
- Babeltrace plugins
- Exports `libbabeltrace`
- Python bindings

Complete environments based on Babeltrace: Trace Compass (Java tool), Eclipse LTTng plugin

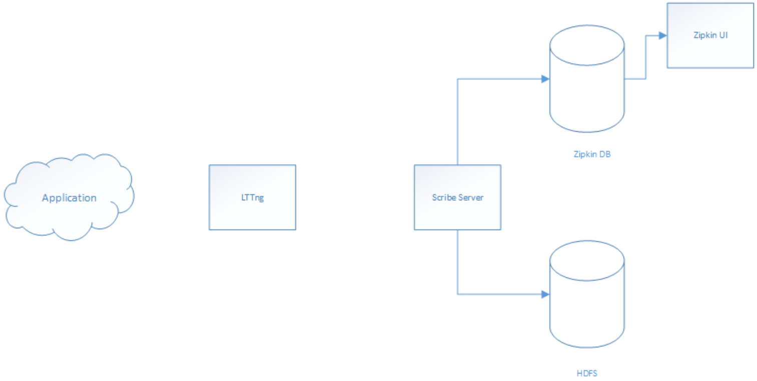
Outline

BlkKin = Block Storage + Zipkin

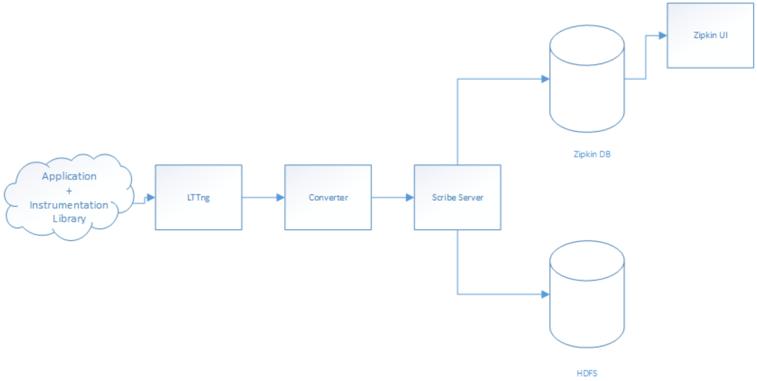
BlkKin is:

- end-to-end tracing infrastructure
- implementing Dapper semantics
- based on Zipkin and LTTng
- providing live tracing

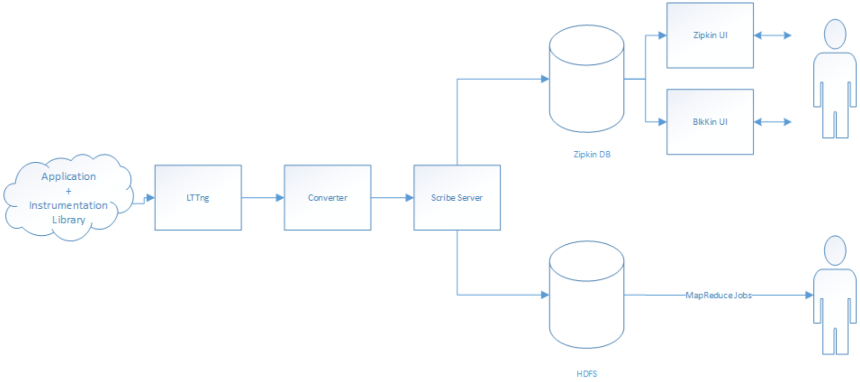
BlkKin Contribution



BlkKin Contribution



BlkKin Contribution



BlkKin Contribution

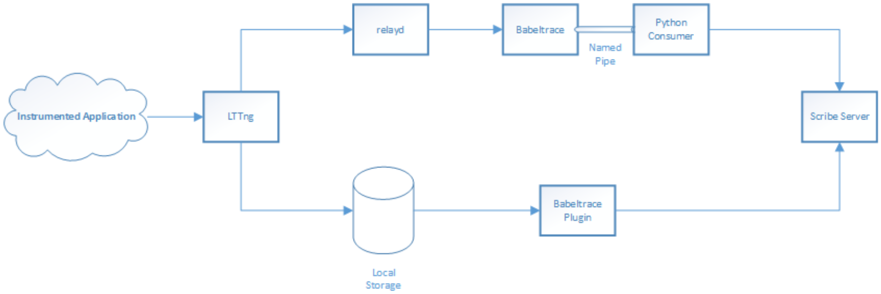
BlkKin instrumentation library:

- C/C++ instrumentation library
- implementing Dapper tracing semantics
- LTTng backend
- sampling

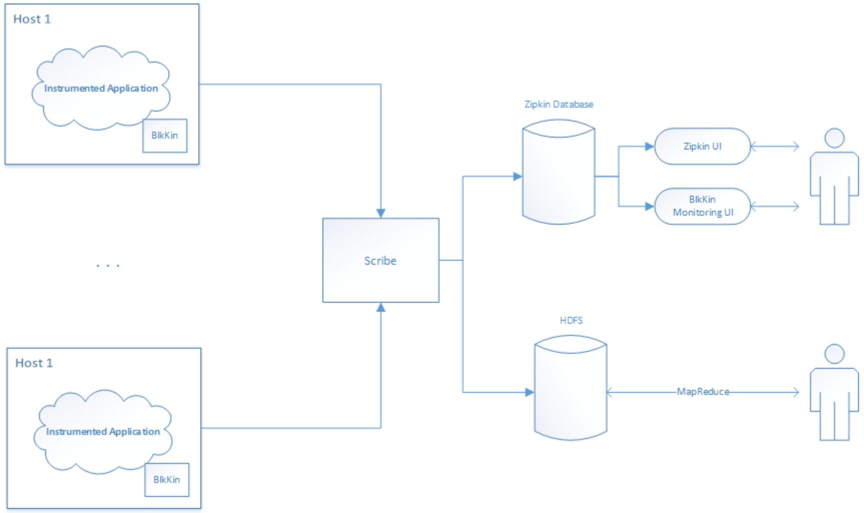
CTF-to-Scribe Babeltrace plugins:

- based on Python bindings
- two output formats:
    - JSON format (generic)
    - Zipkin Thrift (Zipkin specific)

BlkKin Architecture



BlkKin Architecture



Outline

Evaluation Environment

Instrumented:

- QEMU Archipelago driver
- Archipelago and libxseg
- RADOS

Testbed:

- 2 hosts interconnected over LAN
- 2 OSDs per host
- 1 host running QEMU and Archipelago

Zipkin in Action

Zipkin Demo

I/O Loads - Scenarios

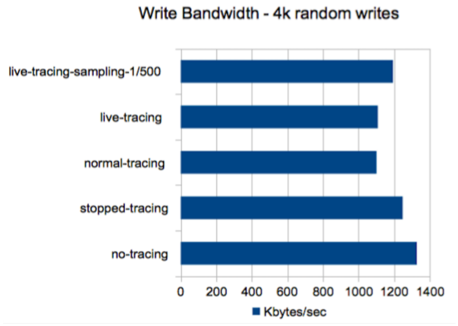
I/O loads created within VM using `fio`:

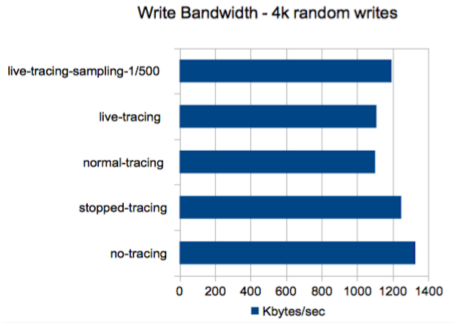
- 4k random writes
- 64k sequential writes

Scenarios:

- no tracing
- stopped tracing
- normal tracing
- live tracing without sampling
- live tracing with 1/500 sampling

Bandwidth Overhead

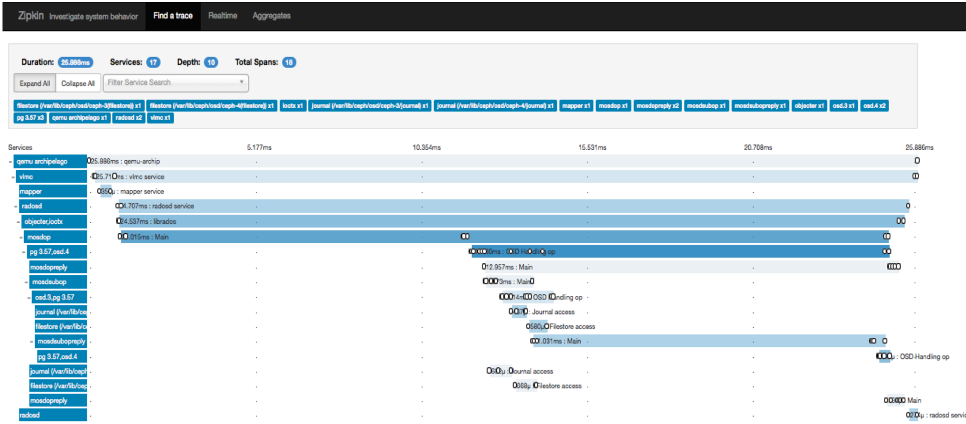




Fault injection

1. Calculate threshold values using MapReduce in HDFS
2. Simulate faulty environment:
   - Network faults: using `tc`
   - Disk faults: adding extra I/O load using `fio`
3. Detect faults using BlkKin monitoring UI

Network fault

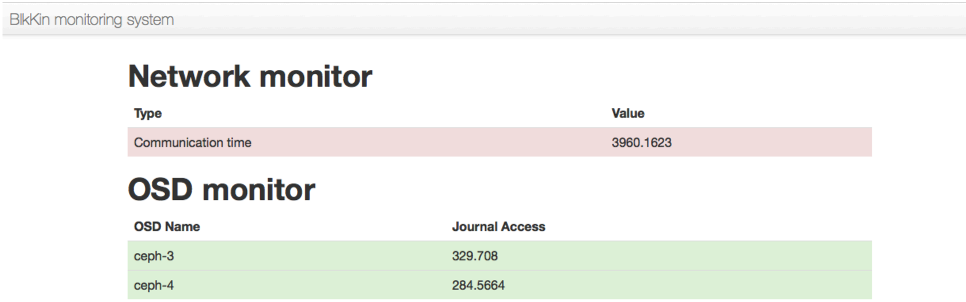


Duration: 25.886ms Services: 17 Depth: 10 Total Spans: 18

filestore (/var/lib/ceph/osd/ceph-3(filestore)) x1 | filestore (/var/lib/ceph/osd/ceph-4(filestore)) x1 | ioctx x1 | journal (/var/lib/ceph/osd/ceph-3/journal) x1 | journal (/var/lib/ceph/osd/ceph-4/journal) x1 | mapper x1 | mosdop x1 | mosdopreply x2 | mosdsubop x1 | mosdsubopreply x1 | objecter x1 | osd.3 x1 | osd.4 x2 | pg 3.57 x3 | qemu archipelago x1 | radosd x2 | vlmc x1

```
Services                      5.177ms     10.354ms     15.531ms     20.708ms     25.886ms
qemu archipelago      25.886ms : qemu-archip
vlmc                  25.71 ms : vlmc service
mapper                035 μ : mapper service
radosd                4.707ms : radosd service
objecter,ioctx        24.537ms : librados
mosdop                .015ms : Main
pg 3.57,osd.4         0ms : OSD Handling op
mosdopreply           12.957ms : Main
mosdsubop             3ms : Main
osd.3,pg 3.57         14m OSD ndling op
journal (/var/lib/ce  : Journal access
filestore (/var/lib/c 580μ Filestore access
mosdsubopreply        1.031ms : Main
pg 3.57,osd.4         μ : OSD Handling op
journal (/var/lib/ceph 60μ : ournal access
filestore (/var/lib/cep 868μ Filestore access
mosdopreply           Main
radosd                μ : radosd servic
```

Network fault



BlkKin monitoring system

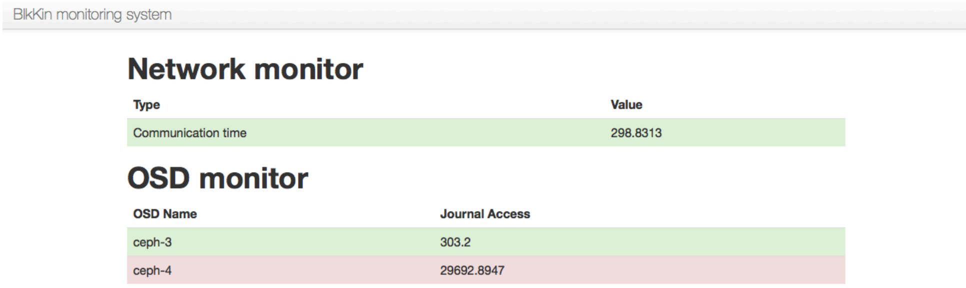
Network monitor

Type	Value
Communication time	3960.1623

OSD monitor

OSD Name	Journal Access
ceph-3	329.708
ceph-4	284.5664

Disk fault



BlkKin monitoring system

Network monitor

Type	Value
Communication time	298.8313

OSD monitor

OSD Name	Journal Access
ceph-3	303.2
ceph-4	29692.8947

Outline

Assumptions

In order to make BlkKin a valuable working system we had to:

- Understand cutting-edge opensource technologies used in the industry
- Combine and extend them
- Use BlkKin in a production scale system, Archipelago and RADOS

Future Plans

- Use BlkKin as official RADOS tracing infrastructure
- Use BlkKin in other low-overhead systems
- Merge Babeltrace plugins in the LTTng source tree

Questions?

"Judge a man by his questions rather than by his answers."

—Voltaire

Thank you