类DAPPER系统

技术，公司内外现状分析——BY 刘卓

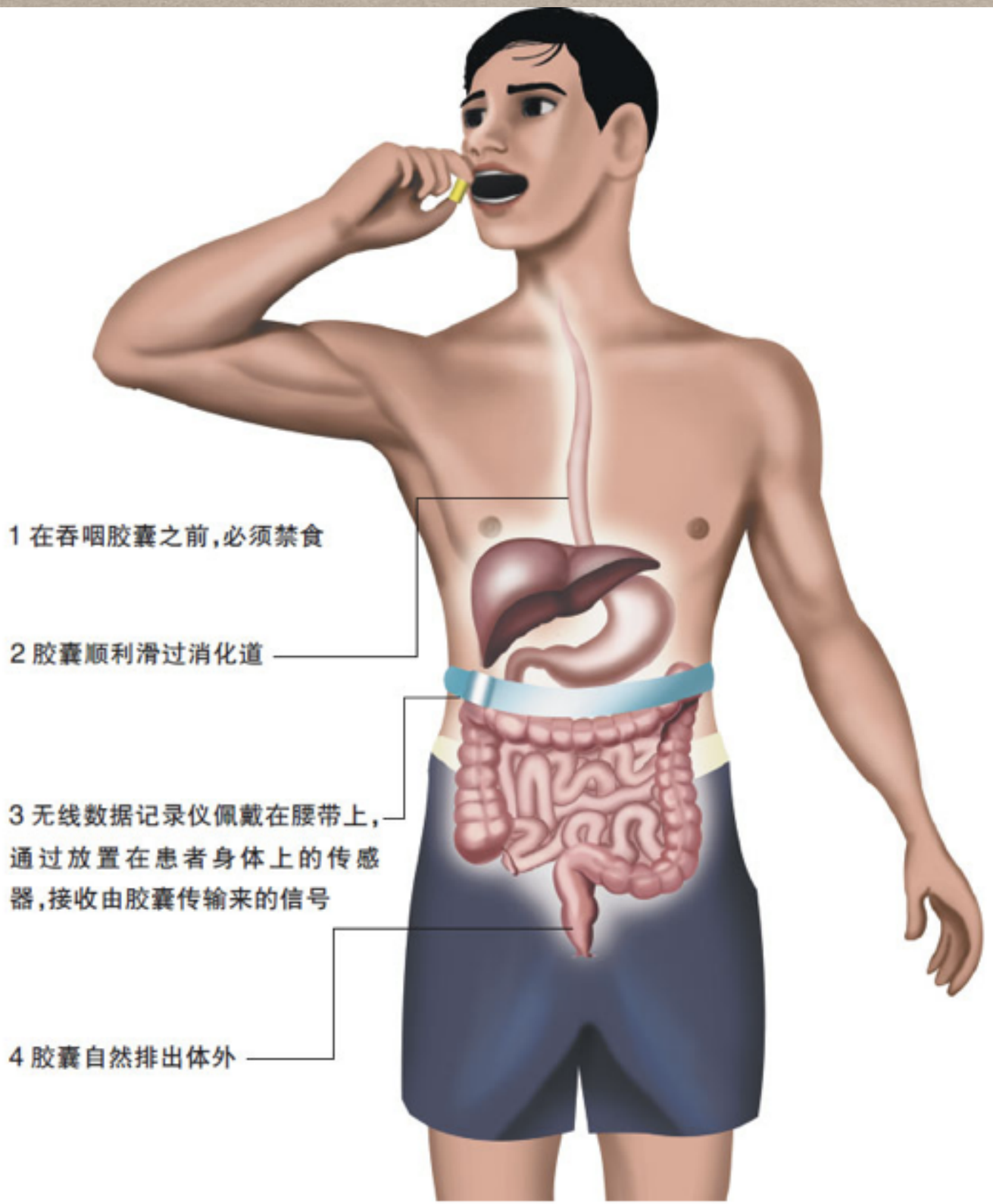
AGENDA

- dapper背景和原理
- dapper应用场景
- 鹰眼的改进
- 业界同类产品技术架构
- baidu同类产品的做法和架构
- baidu rpc技术现状
- 当下的考虑

AGENDA

- dapper背景和原理
- dapper应用场景
- 鹰眼的改进
- 业界同类产品技术架构
- baidu同类产品的做法和架构
- baidu rpc技术现状
- 当下的考虑

胶囊内镜



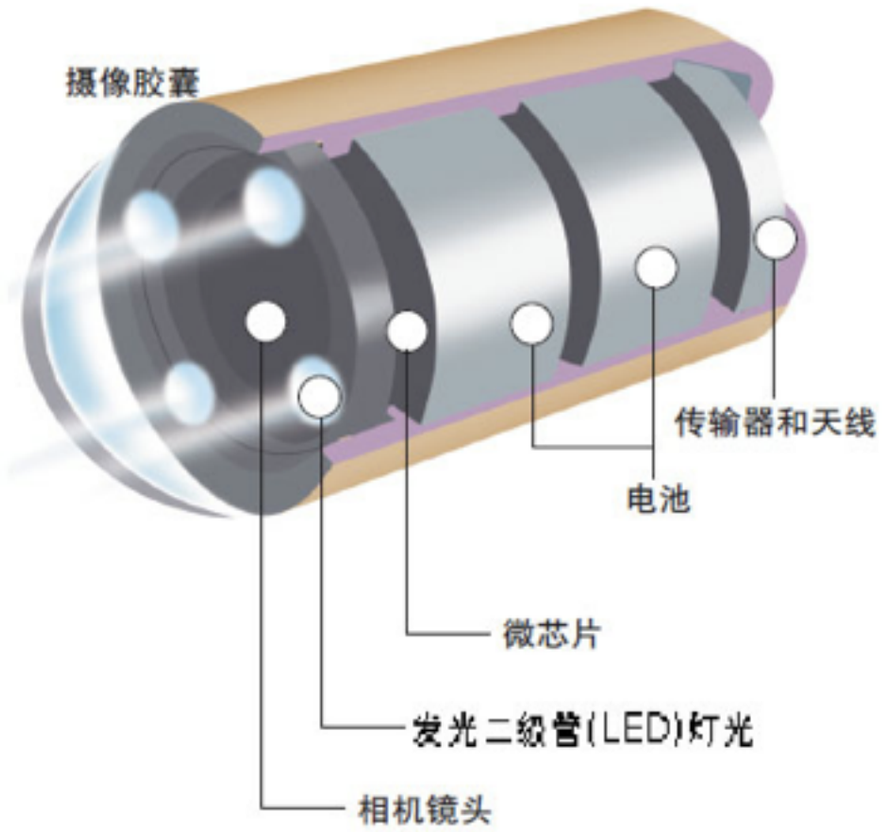


图 胶囊内镜系统

胶囊装配有一个一次性的微小摄像机，能够对标准内镜无法到达的小肠部分进行检查，以诊断不明原因消化道出血或者其他异常。视频资料被传输和储存在一个数据记录仪内，该记录仪佩戴在腰带上。之后，资料被下载到工作站的计算机中，供医师分析。

DAPPER产生的背景

- universal search(全局搜索）
- 对搜索耗时细节的了解

GOOGLE面对的困难

- 搜索调用了那些服务（服务，功能频繁上下线，导致不可认知）
- 一个工程师无法对所有服务都了解（每个服务由不同团队维护）
- 后台系统同时对多个应用提供服务，问题可能有其他的应用造成（平台类服务）

DAPPER设计目标

- ubiquitous deployment（无处不在的部署）
- continuous monitoring（不间断的监控）

DAPPER REQUIREMENTS

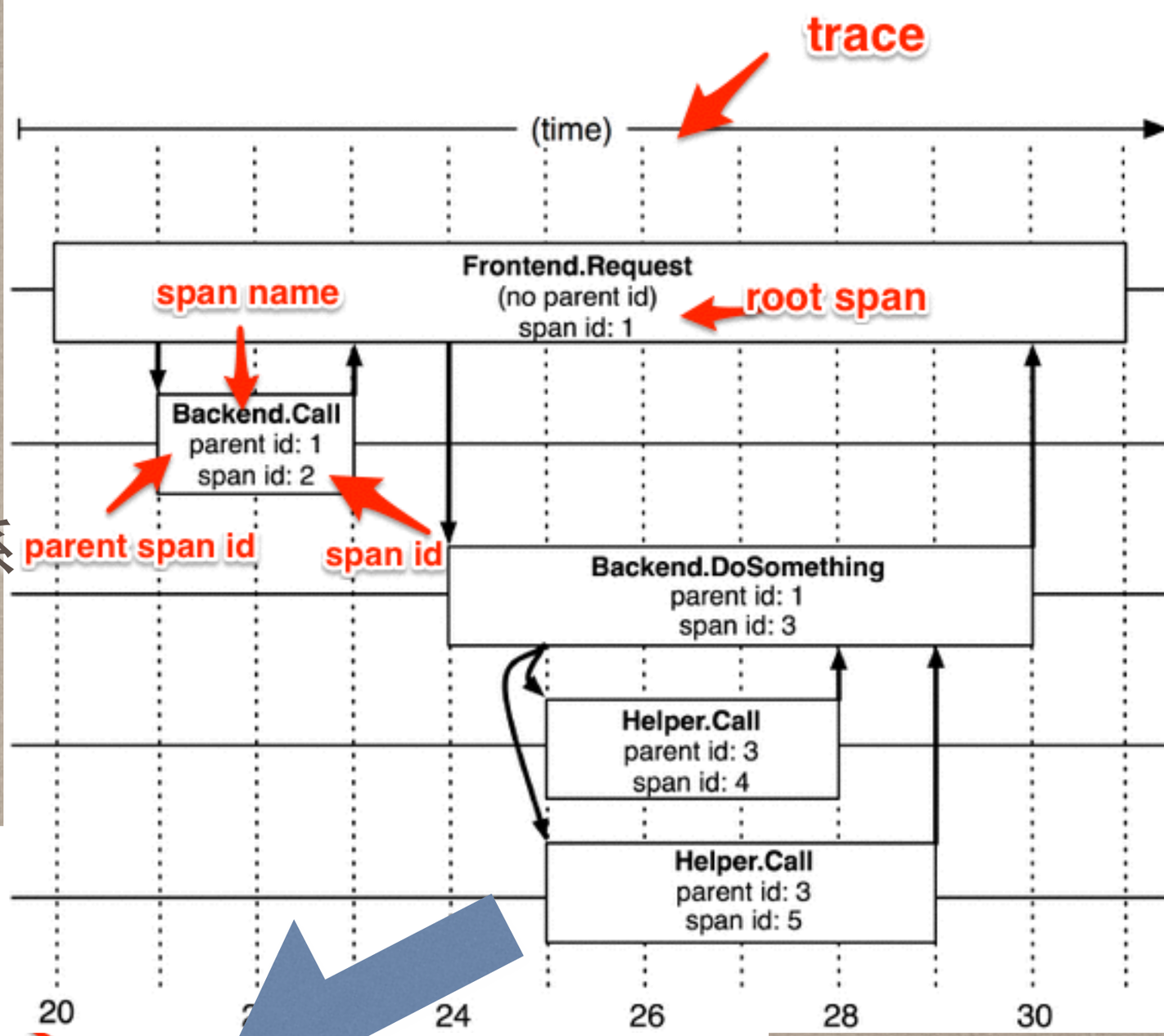
- Low overhead
- Application-level transparency
- Scalability
- Tracing data to be available for analysis quickly(< 1 minute)

关键技术架构

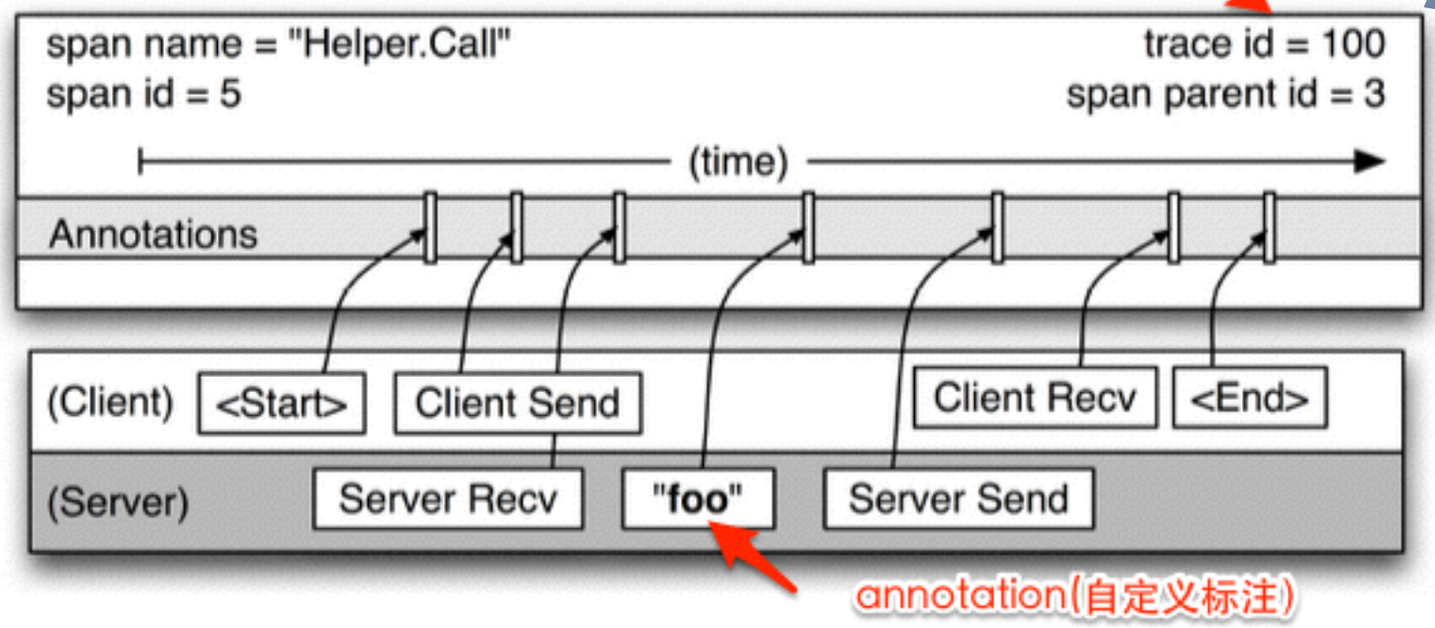
- rpc植入
- 日志采集，存储，API，可视化

RPC植入

植入了性能相关的数据
植入了*rpc*依赖关系
植入了*trace*和*span*的归属关系
提供自定义信息植入的接口



每个span中包含全局唯一的trace id



采样率

- 1/1024
- 自适应采样率（依据负载来调整）
- 在存储端做二次采样（控制集群数据规模和吞吐）

数据规模

- 每天1TB数据
- 保留2周
- 共14TB的数据量

数据可视化

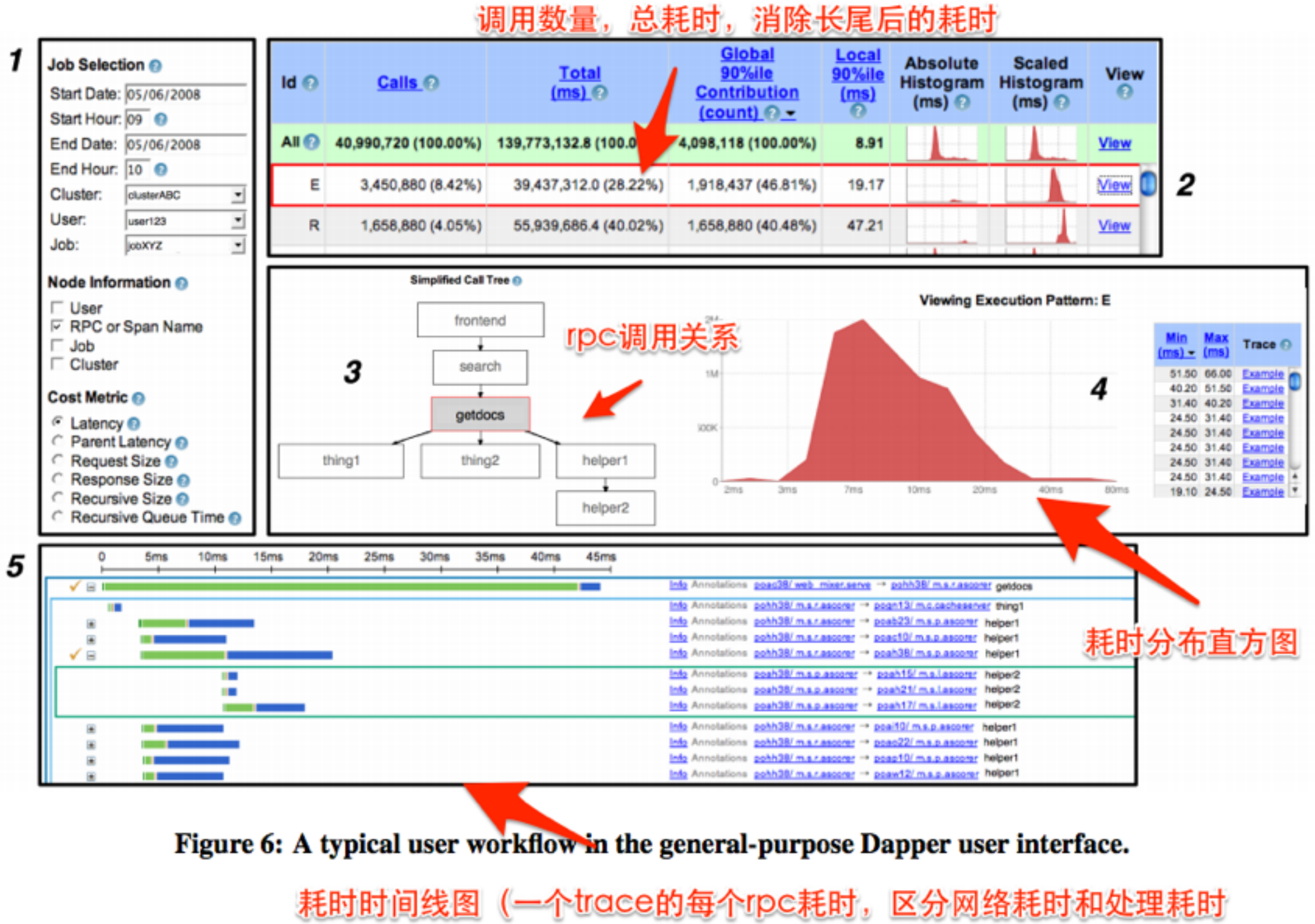


Figure 6: A typical user workflow in the general-purpose Dapper user interface.

耗时时间线图（一个trace的每个rpc耗时，区分网络耗时和处理耗时

AGENDA

- dapper背景和原理
- dapper应用场景
- 鹰眼的改进
- 业界同类产品技术架构
- baidu同类产品的做法和架构
- baidu rpc技术现状
- 当下的考虑

应用场景

- AdWords
- 解决延迟的长尾效应
- 推断服务依赖
- 不同服务的网络使用率
- 分层和共享存储系统
- 用dapper救火
- 安全领域

场景1：ADWORDS

- 性能：定位延迟的关键环节；确定哪些请求是不必要的串行请求
- 正确性：哪些策略被访问，比如：哪些访问主库，哪些访问副本库
- 可理解性：确定总查询成本；促进对系统的重新设计
- 测试：QA利用dapper验证正确的系统行为和性能
- 结合监控系统异常请求的上下文定位问题
- 服务延迟优化了2个数量级

场景2：解决延迟的长尾效应

- 判断搜索请求的关键路径
- 找到意想不到的服务之间的交互
- 建立在每个sub system中都慢的query 列表
- 结论：关键路径的网络性能退化不影响系统的吞吐，但影响延迟。

场景3：推断服务依赖

- 不同cluster中job的依赖关系
- 标注bigtable的table name，用于在服务粒度上判断依赖关系

场景4：不同服务的网络使用率

- 从应用角度找到网络流量的热点
- 从应用角度比从机器角度能获得更有价值的信息

场景5：分层和共享存储系统

- App Engine -> BigTable -> GFS
  - -> Chubby
- 为平台类服务提供不同维度的用户排名（网络负载维度，服务请求耗用的总时间）

场景6：用DAPPER救火

- 快速获取数据，在救火的时候做参考

场景7：安全领域

- 审核安全参数
- 确保不能向未授权系统发送请求

AGENDA

- dapper背景和原理
- dapper应用场景
- 鹰眼的改进
- 业界同类产品技术架构
- baidu同类产品的做法和架构
- baidu rpc技术现状
- 我的考虑

鹰眼的改进

- 在在线服务的基础上结合了离线计算。

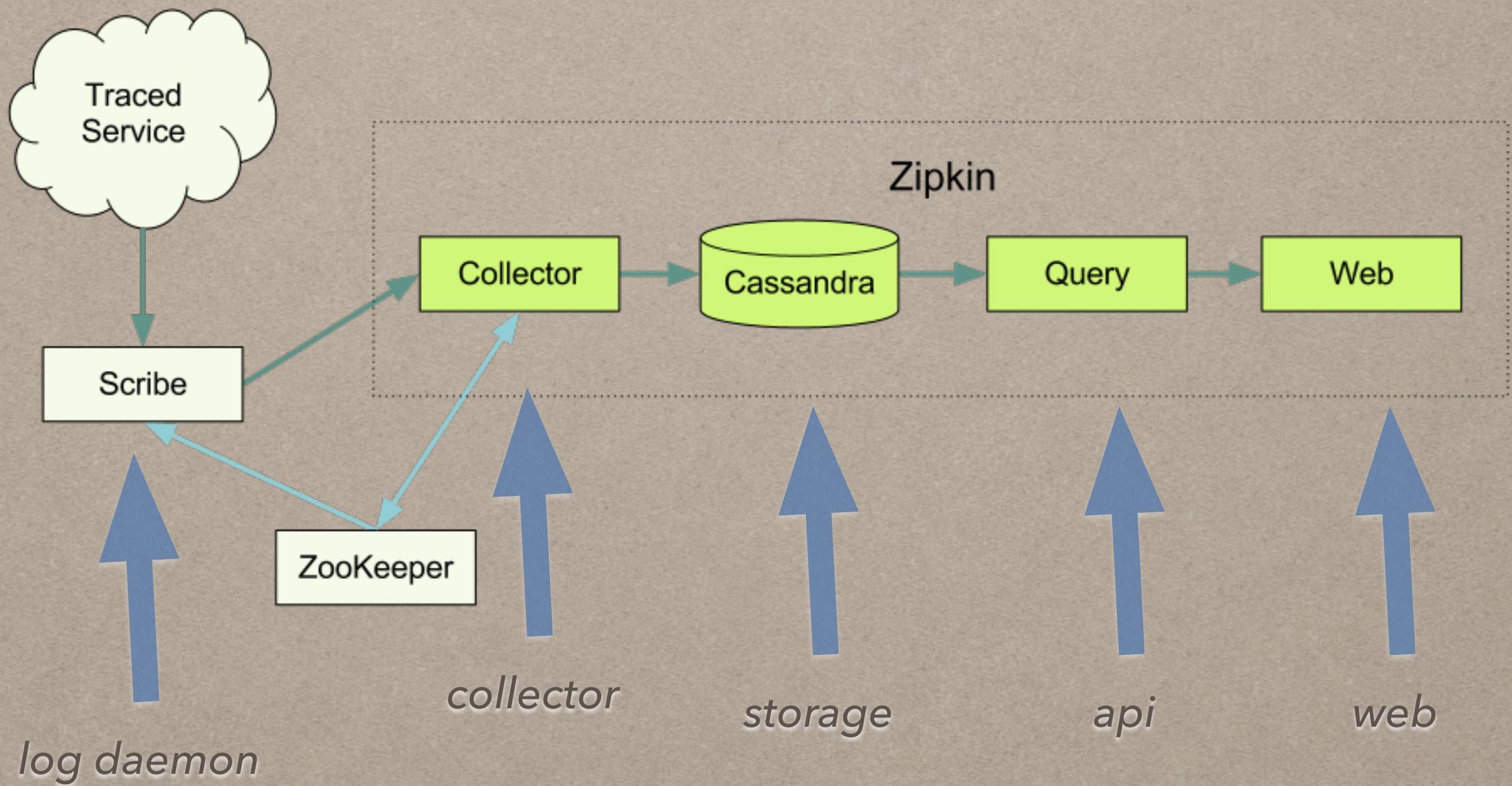
鹰眼的应用场景（没有超越DAPPER论文中的场景）

- 单条调用链（请求）。主要结合异常堆栈，辅助cpu，jvm，io，网络定位问题。
- 多条调用链做统计分析。
    - 容量规划。（平台类服务调用量增长）
    - 调用来源分析。（整体流量突增由哪个请求来源造成）
    - 依赖度量。（强依赖，依赖几率，依赖频度）
    - 调用耗时。（判断瓶颈点）
    - 调用并行度。（用于做异步优化的依据）
    - 调用路由分析。（热点分析，检验路由状况）
- 人为标注

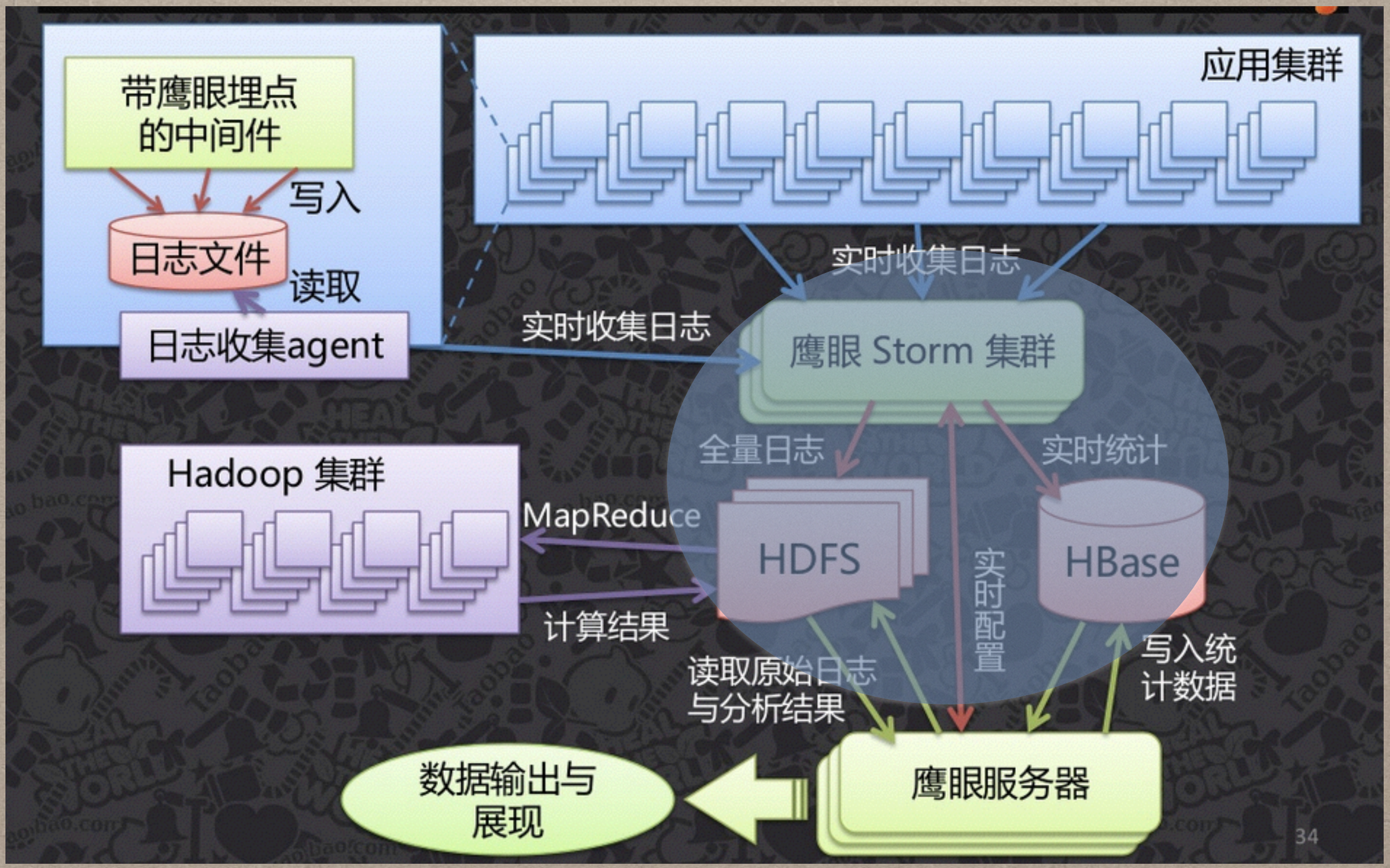
AGENDA

- dapper背景和原理
- dapper应用场景
- 鹰眼的改进
- 业界同类产品技术架构
- baidu同类产品的做法和架构
- baidu rpc技术现状
- 当下的考虑

ZIPKIN


Traced Service
Zipkin
Collector
Cassandra
Query
Web
Scribe
ZooKeeper
log daemon
collector
storage
api
web

鹰眼


带鹰眼埋点
的中间件
写入
日志文件
读取
日志收集agent
应用集群
实时收集日志
实时收集日志
鹰眼 Storm 集群
全量日志
实时统计
Hadoop 集群
MapReduce
HDFS
实时配置
HBase
计算结果
读取原始日志
与分析结果
写入统
计数据
数据输出与
展现
鹰眼服务器