SACC 2016 第八届中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2016
架构创新之路

图数据库 Neo4j 的实践之路

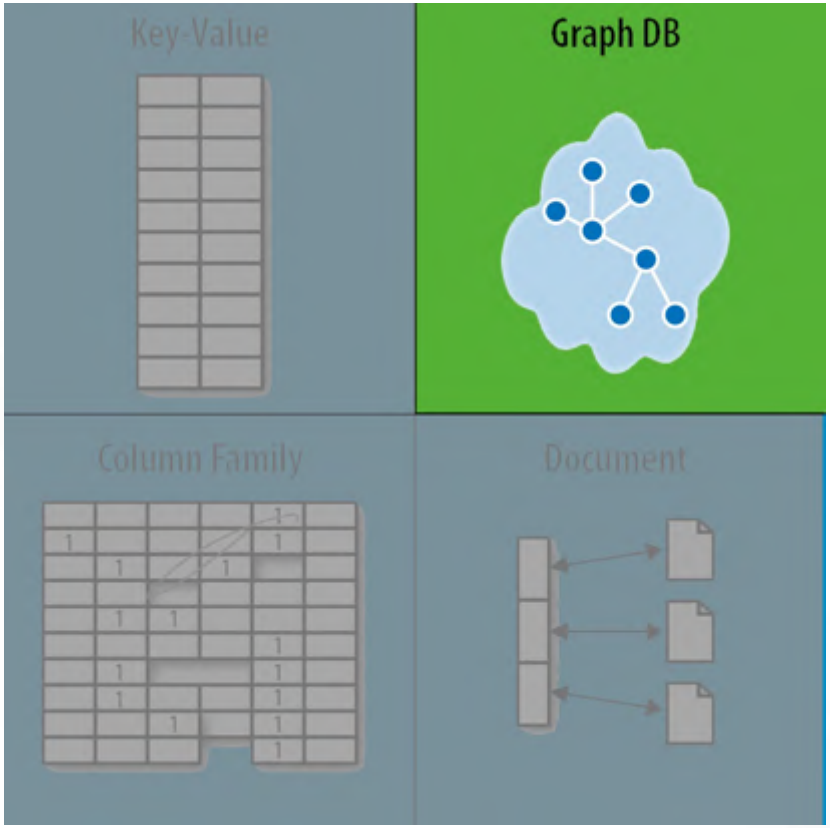
魏佳
技术主管 @ LinkedIn中国

关于我

- 应用生命周期管理（ALM）
- 分布式消息中间件
- 虚拟化及多租户运行时（Multi-tenant runtime）
- 分布式系统请求链路追踪和服务质量（SLA）
- 著有多篇国内外技术专利

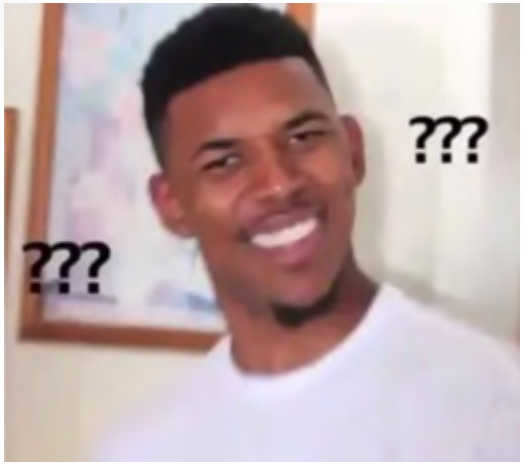
关于此分享

- 内容
- 受众



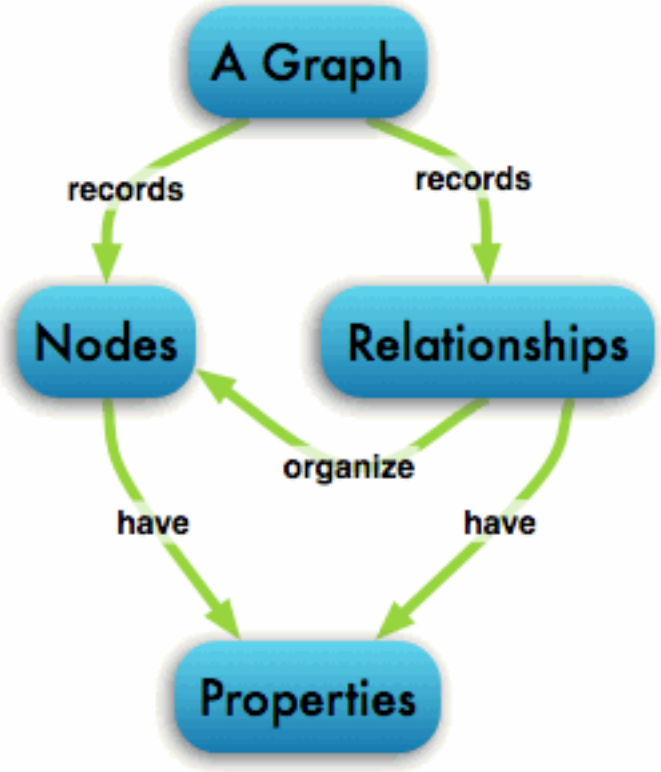
图数据库

“...使用图结构（graph structure）来实现语义查询，使用点（vertex）、边（edge）和属性（property）来表示和存储数据。最重要的概念是边（关系），在存储中，它直接将数据项关联起来...”



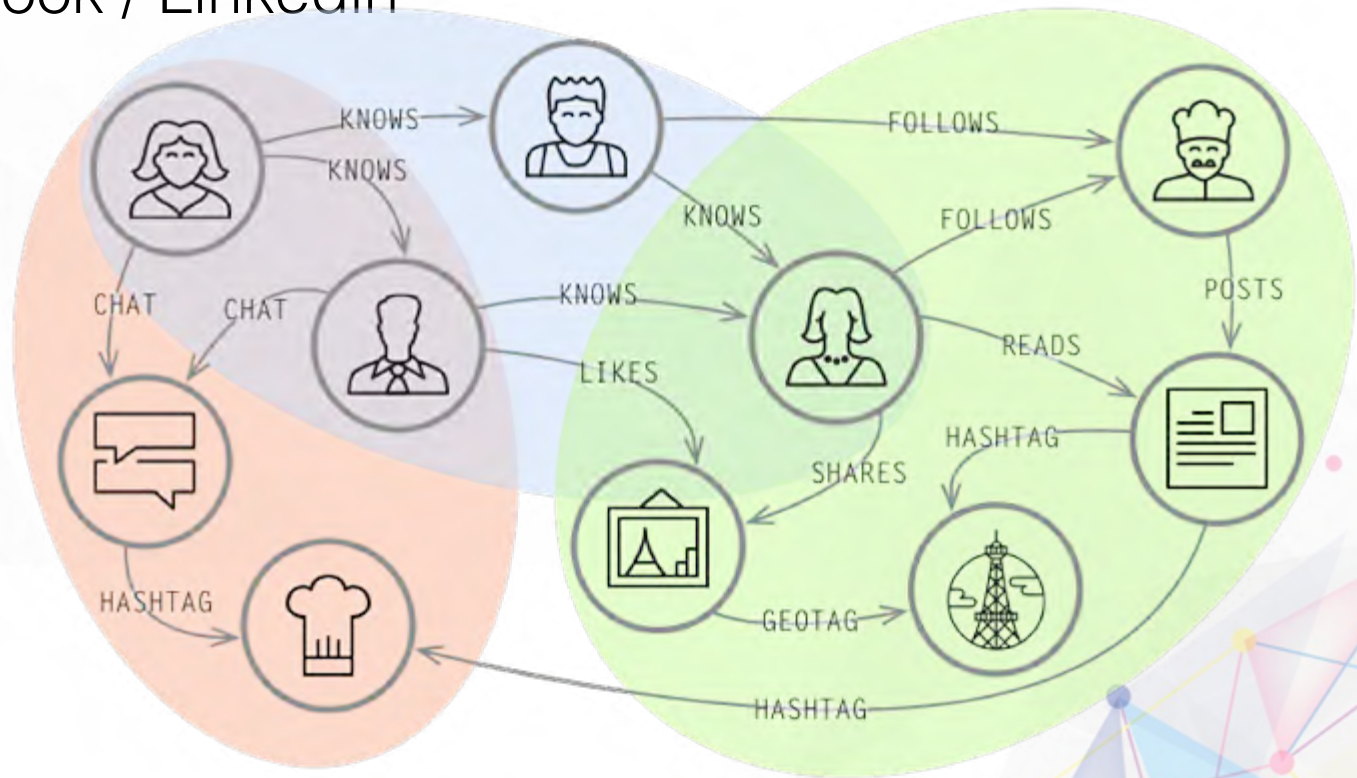
https://en.wikipedia.org/wiki/Graph_database

图数据库


A Graph
records
records
Nodes
Relationships
organize
have
have
Properties

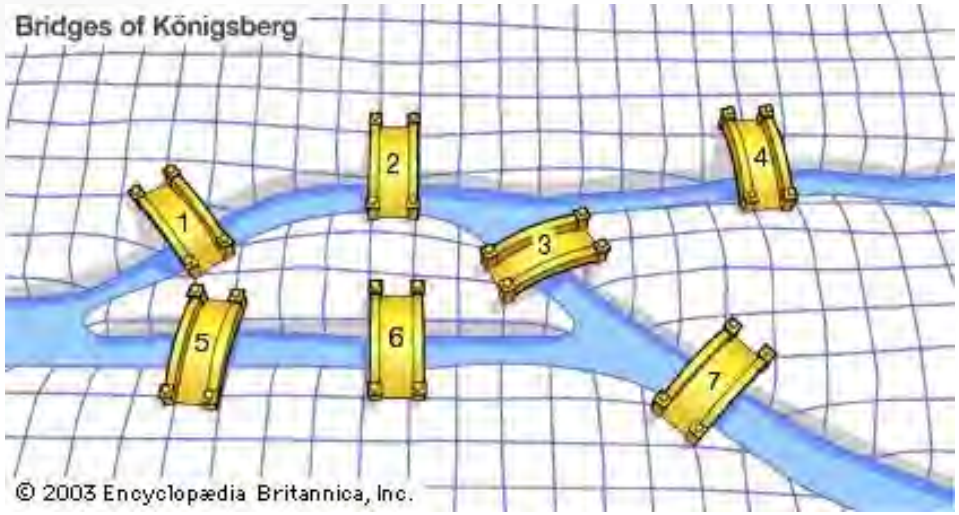
场景

- Social Network
  - Facebook / Linkedin



场景

- Spatial/GIS
  - 路径规划
  - 交通预测



Problem Definition

- The traveling salesman problem consists of a salesman and a set of cities. The salesman has to visit each one of the cities starting from a certain one (e.g. the hometown) and returning to the same city. The challenge of the problem is that the traveling salesman wants to minimize the total length of the trip.

Example

Consider the following set of cities:

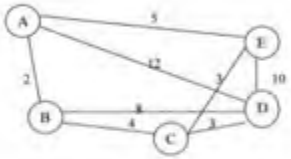


Figure 10.1 A graph with weights on its edges

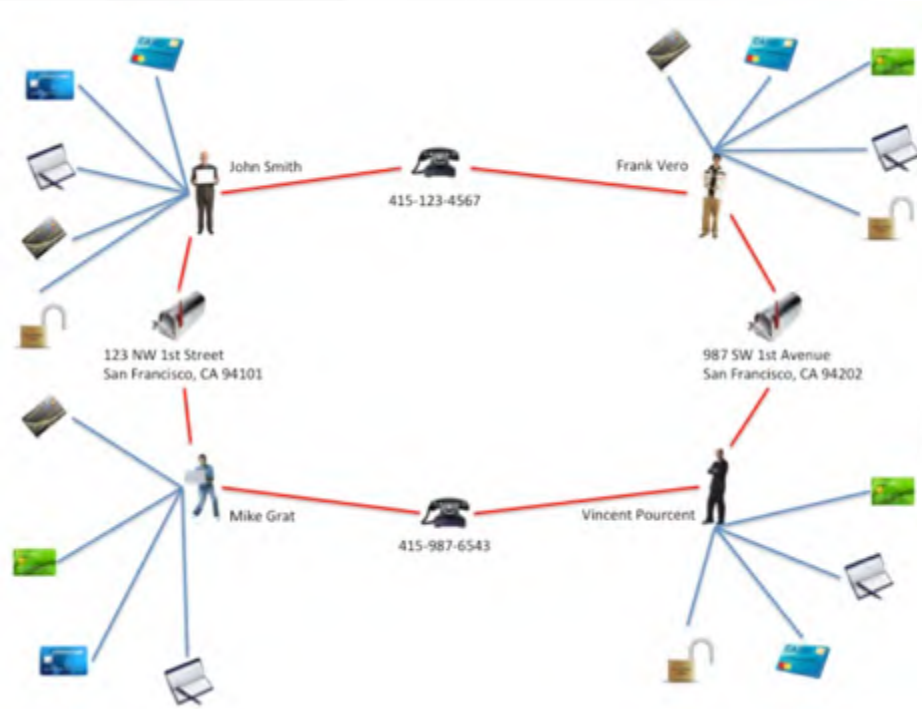
The problem lies in finding a minimal path passing from all vertices once. For example the path Path1 {A, B, C, D, E, A} and the path Path2 {A, B, C, E, D, A} pass all the vertices but Path1 has a total length of 24 and Path2 has a total length of 31.

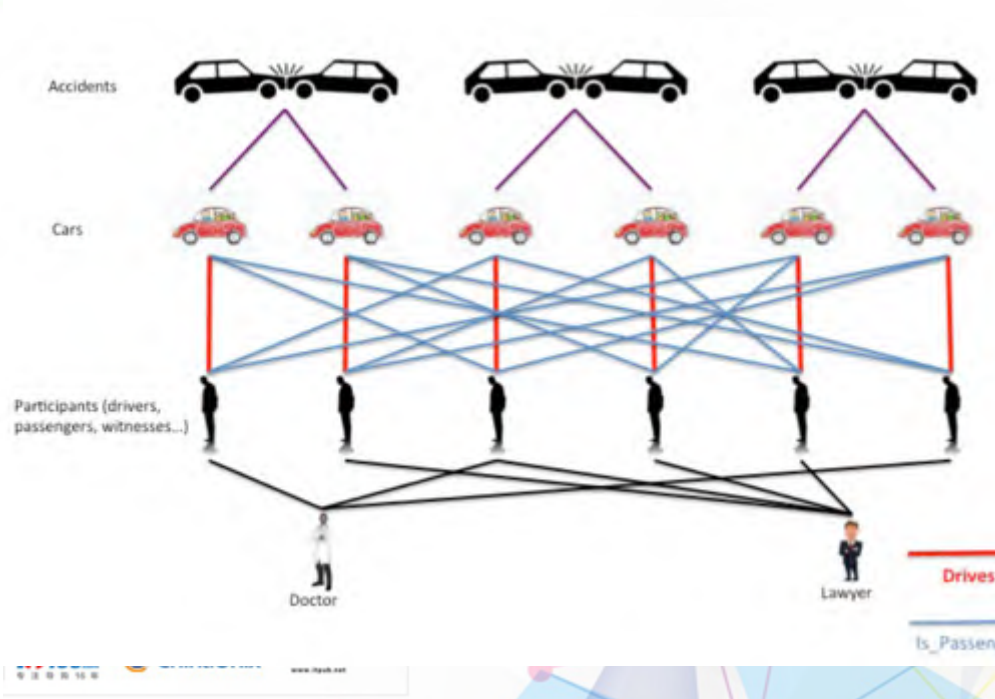
场景

- Recommendation
  - “You may also like…”
  - “People you may know…

场景

- Fraud detection
  - First-party bank fraud/Insurance fraud/e-Commerce fraud





场景

- Workflow engine
- Access control
- …

场景

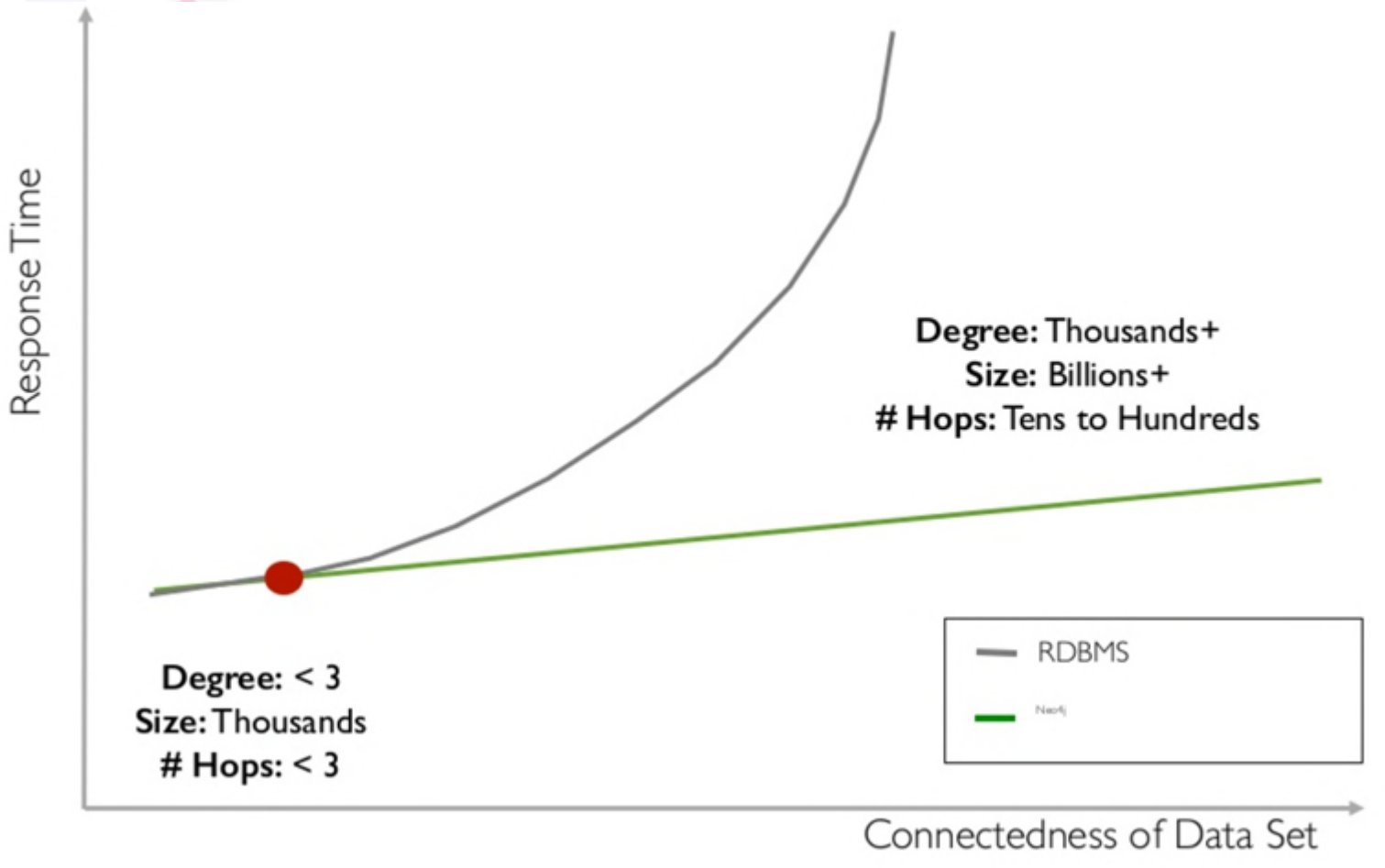
场景

Graph Query Response Time =

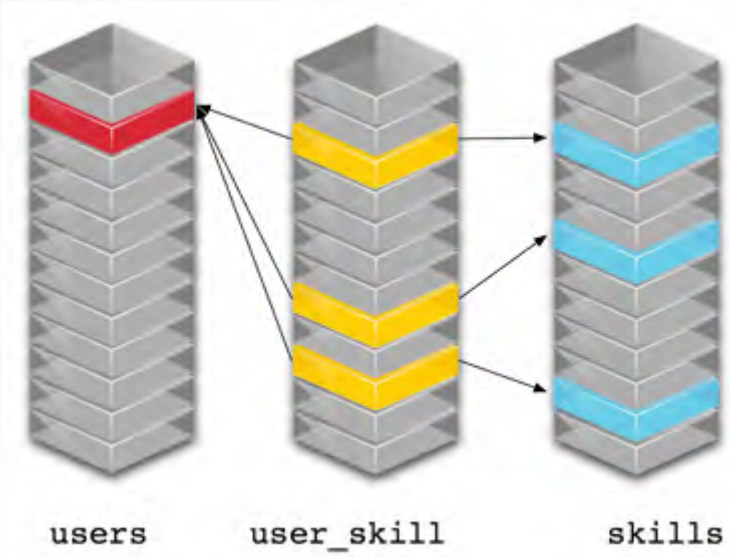
f(graph density, graph size, query degree)

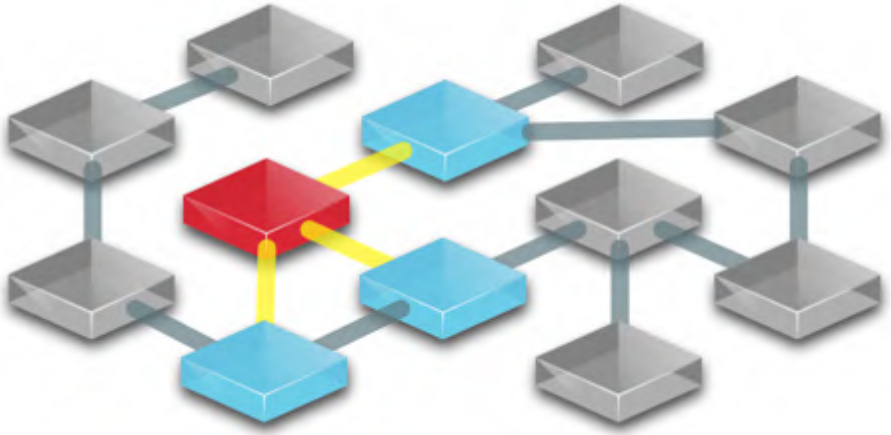
- graph density: avg. (relations per node)
- graph size: ∑(node)
- query degree: hops per query

场景


Response Time
Degree: Thousands+
Size: Billions+
Hops: Tens to Hundreds
Degree: < 3
Size: Thousands
Hops: < 3
RDBMS
Neo4j
Connectedness of Data Set

场景

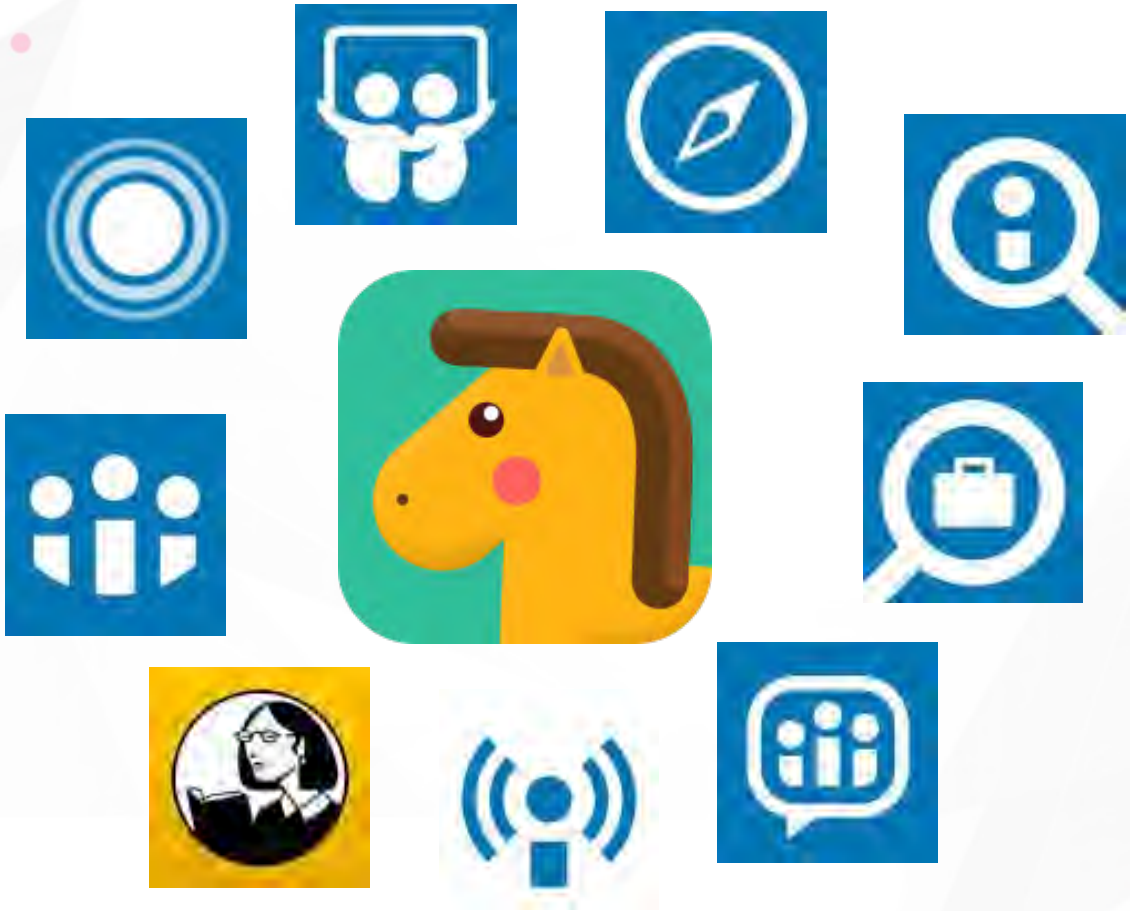




背景

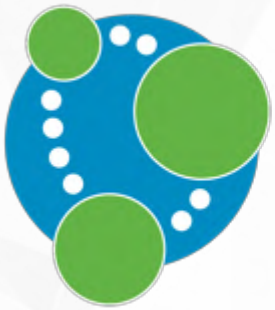
- LinkedIn Flagship（职场人士）
  - LinkedIn Students（学生群体）
  - LinkedIn Influencer（职场影响力内容发布和传播）
  - LinkedIn Job Search
  - LinkedIn Recruiter（企业征才解决方案）
  - LinkedIn Lookup（企业雇员信息解决方案）
- Lynda（全球最大在线职业教育培训）
- Slideshare（全球最大的在线演示文档共享）
- Pulse（阅读和内容聚合）

背景



选型

选型

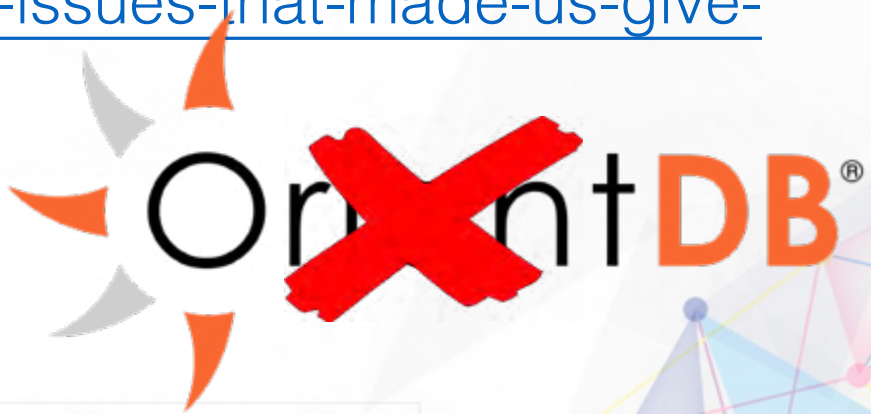


neo4j

TITAN

选型

- OrientDB
  - 缺少批量写，从DB文件加载并初始化非常慢
  - 在200M+节点后写瓶颈
  - 不健壮不稳定
  - CEO不靠谱 http://orientdbleaks.blogspot.com/2015/06/the-orientdb-issues-that-made-us-give-up.html



选型

- Titan
  - 更像是API layer，可选backend engine，Cassandra、HBase、BDB和ElasticSearch
  - ACID不完备，依赖backend engine
  - Key index使用限制较多
  - 缺乏客户案例
  - Dev freeze

选型

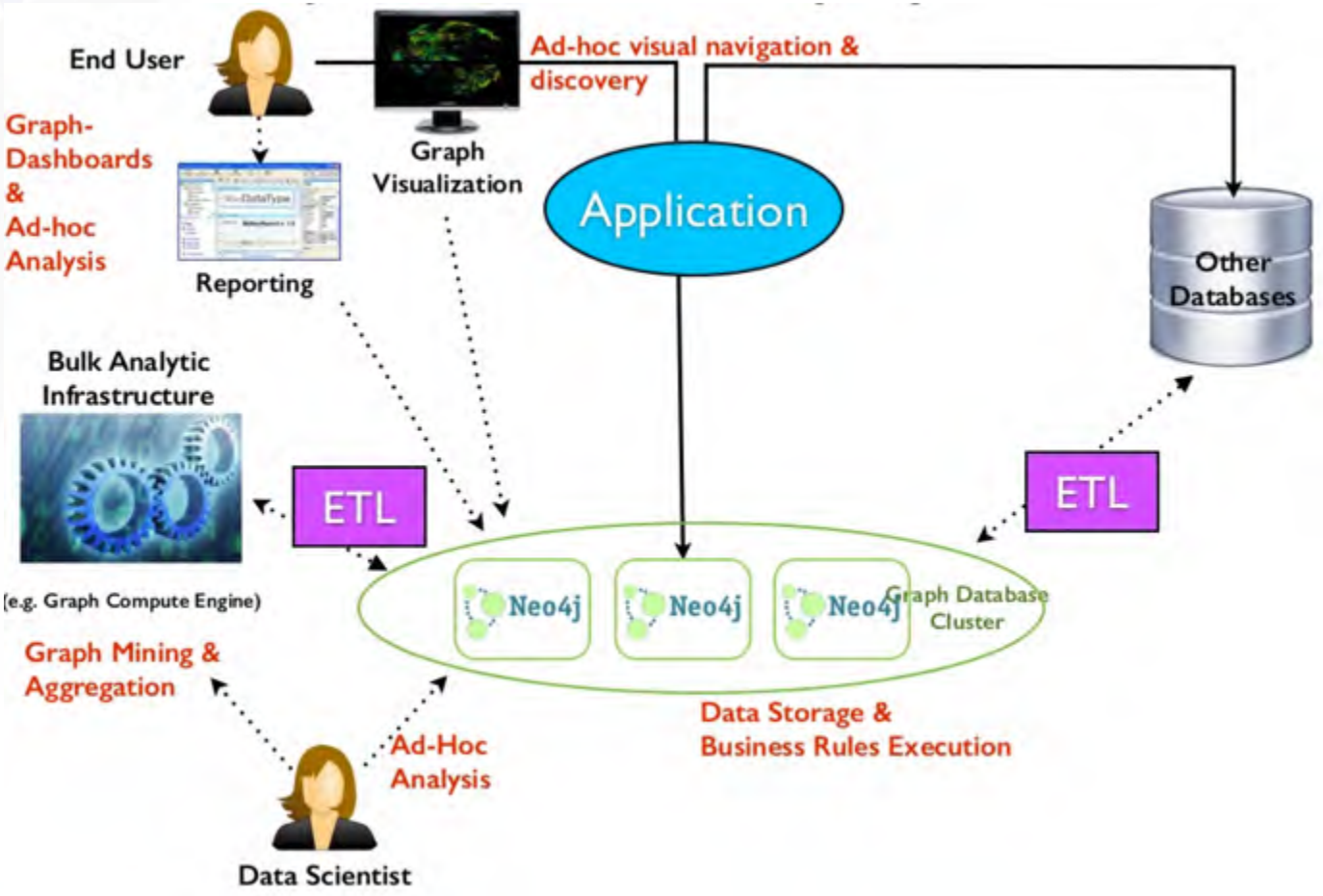
The last man standing, and the winner is

Neo4j

Neo4j

- Schema-free with labeled property
- Fast graph traversal(~2M/s)
- Written in Java
- Full ACID compliant
- Full/incremental online backup
- HTTP API or embedded in JVM
- Powerful declarative query language
- Cache(chunk) based sharding
- Linearly scalable read, M-S replication, fault-tolerance
- Plugins & extensions
- …

Neo4j


End User
Ad-hoc visual navigation & discovery
Graph-Dashboards & Ad-hoc Analysis
Graph Visualization
Application
Reporting
Other Databases
Bulk Analytic Infrastructure
ETL
ETL
(e.g. Graph Compute Engine)
Neo4j
Neo4j
Neo4j
Graph Database Cluster
Graph Mining & Aggregation
Ad-Hoc Analysis
Data Storage & Business Rules Execution
Data Scientist

Neo4j

- Distributed
  - replication
  - fault tolerance
  - scalability
  - consistency

Neo4j

- Cluster Management
  - member join/leave/heartbeat
- Failover
  - election
  - status propagation
- Replication
  - distributed lock
  - transaction pull/push

Apache Zookeeper

HELIX

Neo4j

- Multi-Paxos algorithm
  - less complex and testable
- Ready for cluster Management and election

Neo4j

```
public enum AcceptorState
        implements State<AcceptorContext, AcceptorMessage> {
start {
    public AcceptorState handle(AcceptorContext context, Message<AcceptorMessage> message, MessageHolder outgoing)
        switch (message.getMessageType()) {
            case join: {
                return acceptor;
            }
        }
        return this;
    }
},
acceptor {
    public AcceptorState handle(AcceptorContext context, Message<AcceptorMessage> message, MessageHolder outgoing)
        switch (message.getMessageType()) {
            case prepare: {...}

            case accept: {...}

            case leave: {
                context.leave();
                return start;
            }
        }
        return this;
    }
}
```

new state

no state change for fallthrough

receiver of outgoing messages

self == current state

mutable data

message w/ type & payload

actions

Neo4j

```
public enum HeartbeatState
    implements State<HeartbeatContext, HeartbeatMessage> {
start {...},

heartbeat {
    public HeartbeatState handle(HeartbeatContext context, Message<HeartbeatMessage> message, MessageHolder outgoing)
        final ClusterContext clusterContext = context.getClusterContext();
        final LearnerContext learnerContext = context.getLearnerContext();
        switch (message.getMessageType()) {
            case i_am_alive: {...}
            case timed_out: {...}
            case sendHeartbeat: {
                URI to = message.getPayload();

                // Check if this node is no longer a part of the cluster
                if (clusterContext.getConfiguration().getMembers().contains(to)) {
                    // Send heartbeat message to given server
                    outgoing.offer(to(HeartbeatMessage.i_am_alive, to,
                            new HeartbeatMessage.IAmAliveState(clusterContext.getMe()))
                            .setHeader("last-learned", learnerContext.getLastLearnedInstanceId() + ""));

                    // Set new timeout to send heartbeat to this host
                    clusterContext.timeouts.setTimeout(HeartbeatMessage.sendHeartbeat + "_"
                            + to, timeout(HeartbeatMessage.sendHeartbeat, message, to));
                }
                break;
            }
            case reset_send_heartbeat: {...}
            case suspicions: {...}
            case leave: { return start; }
        }
        return this;
    }
```

state machine

state

mutable state

message-payload

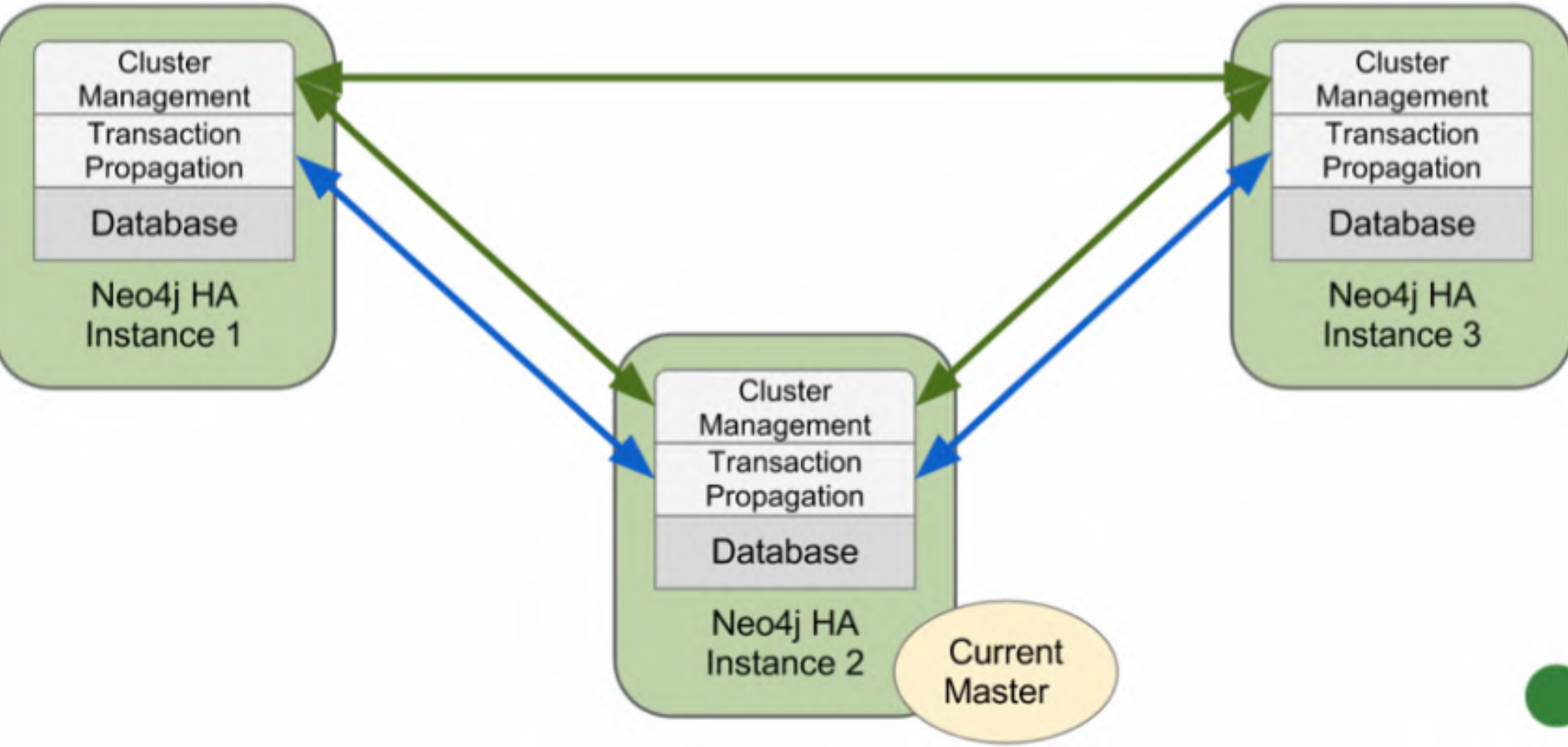
send message

message-payload

timeout with message

new state

Neo4j

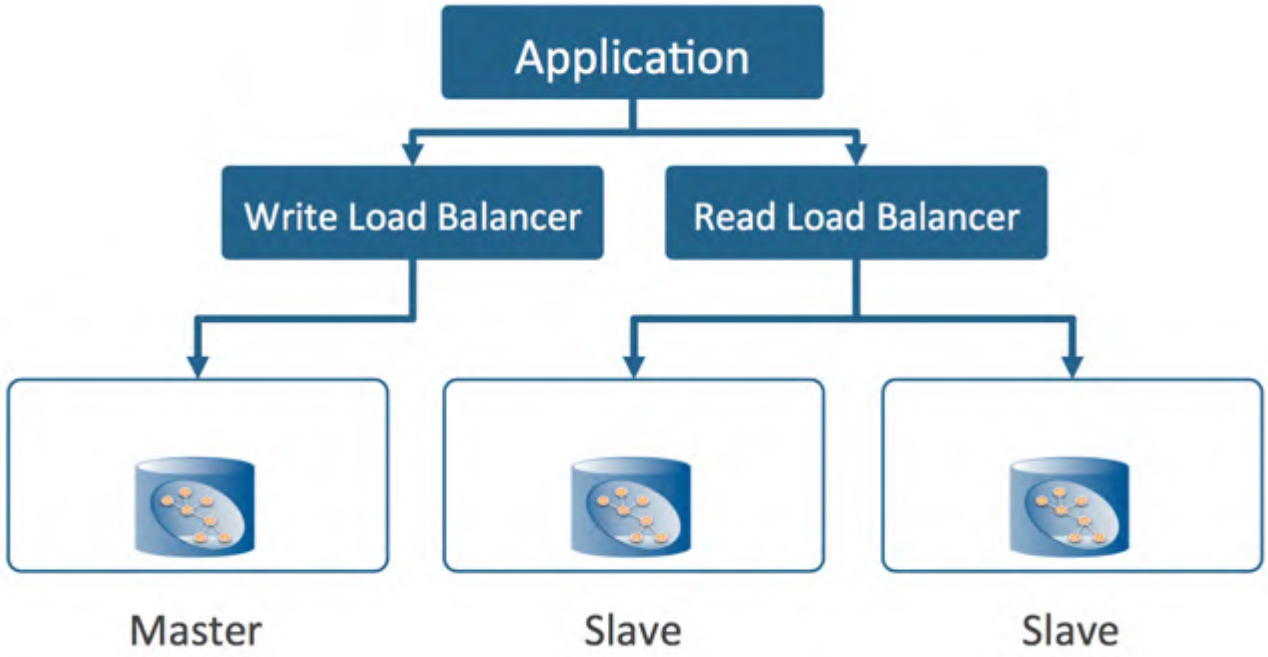

Cluster Management
Transaction Propagation
Database
Neo4j HA Instance 1
Cluster Management
Transaction Propagation
Database
Neo4j HA Instance 3
Cluster Management
Transaction Propagation
Database
Neo4j HA Instance 2
Current Master

Neo4j

- All transactions are committed on the master
  - eventually applied to the slaves
  - eventuality defined by update interval
- When writing on a slave
  - lock coordinated through the master
  - transaction buffered on the slave
  - applied on the master to obtain txid
  - applied to the slaves

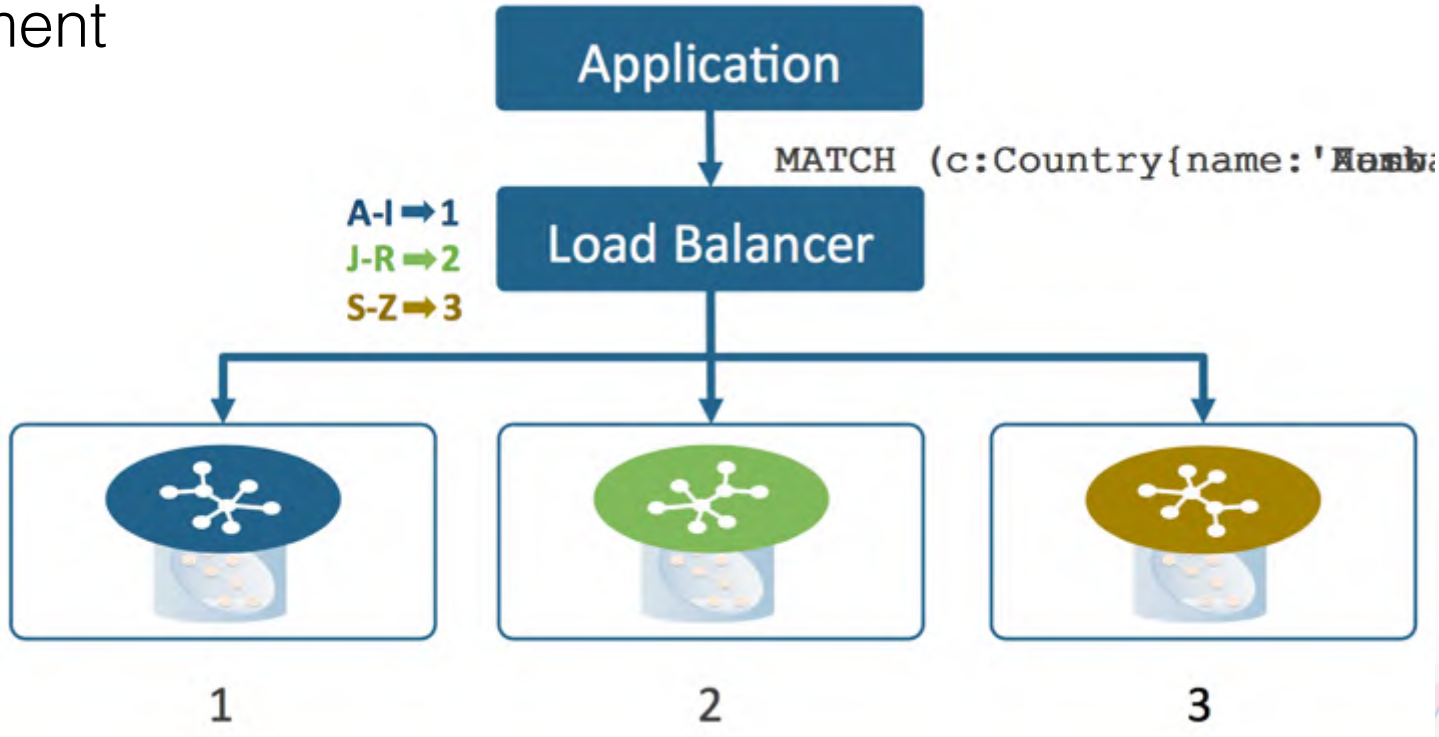
Neo4j

- Deployment



Neo4j

- Deployment



踩坑

踩坑

- GC暂停导致重选举
  - 进而一致性检查，引起数据复制，最终造成若干节点不可用
- JSON payload大，影响吞吐同时浪费带宽
  - 同时heap中产生大量脏footprint，对GC不利
- HTTP based API效率不高
  - 短链接，协议处理的开销
- 复杂查询场景效率不高
  - 依赖外部其他资源时，结果集利用率低
  - 多次请求时，不必要的网络延时

踩坑

- GC暂停导致重选举
  - 进而一致性检查，引起数据复制，最终造成若干节点不可用
- JSON payload大，影响吞吐同时浪费带宽
  - 同时heap中产生大量脏footprint，对GC不利
- HTTP based API效率不高
  - 短链接，协议处理的开销
- 复杂查询场景效率不高
  - 依赖外部其他资源，结果集利用率低
  - 多次请求，不必要的网络延时

改造和优化

- GC调优
  - G1收集器
    - adaptive
  - JFR+JMC
    - on-flight
    - visualized
    - insignificant performance overhead

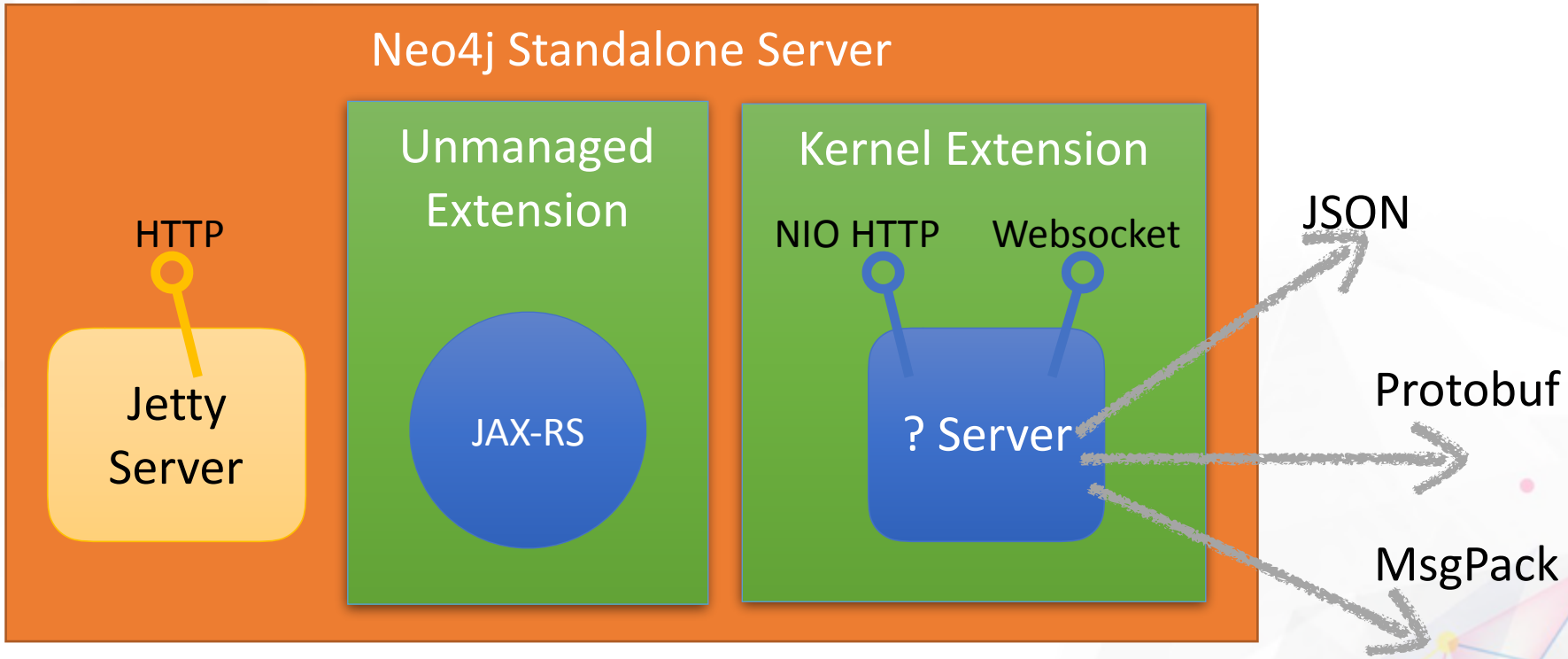
踩坑

- GC暂停导致重选举
  - 进而一致性检查，引起数据复制，最终造成若干节点不可用
- JSON payload大，影响吞吐同时浪费带宽
  - 同时heap中产生大量脏footprint，对GC不利
- HTTP based API效率不高
  - 短链接，协议处理的开销
- 复杂查询场景效率不高
  - 依赖外部其他资源，结果集利用率低
  - 多次请求，不必要的网络延时

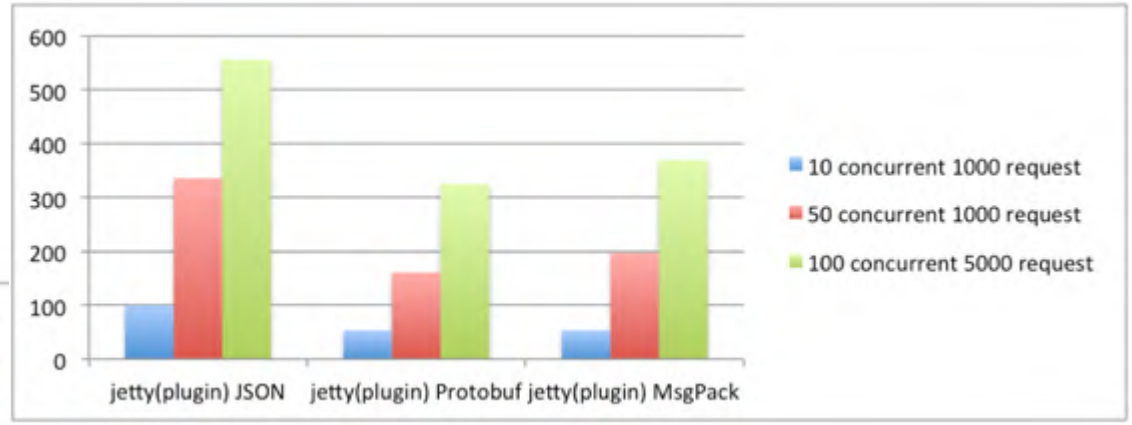
改造和优化

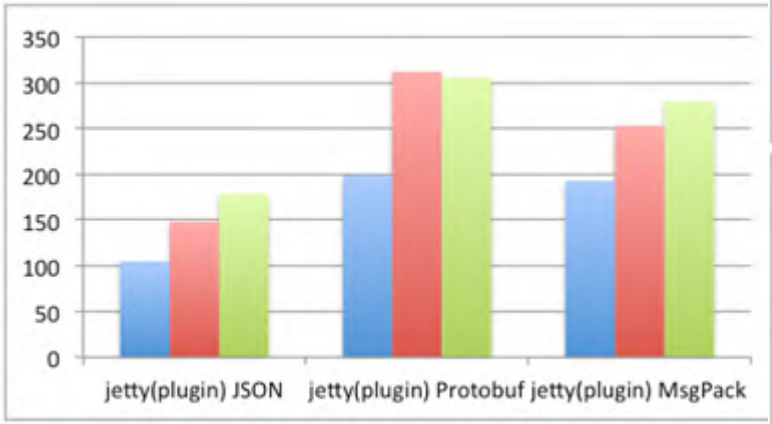
- 协议，传输
  - Jetty+Jersey serve HTTP request/response
  - poor performance
- 数据
  - JSON（jackson streaming）
  - serialization overhead
- Plugin/Extension
  - ~~Plugin~~
  - Unmanaged extension（JAX-RS）
  - Kernel extension（no constraint）

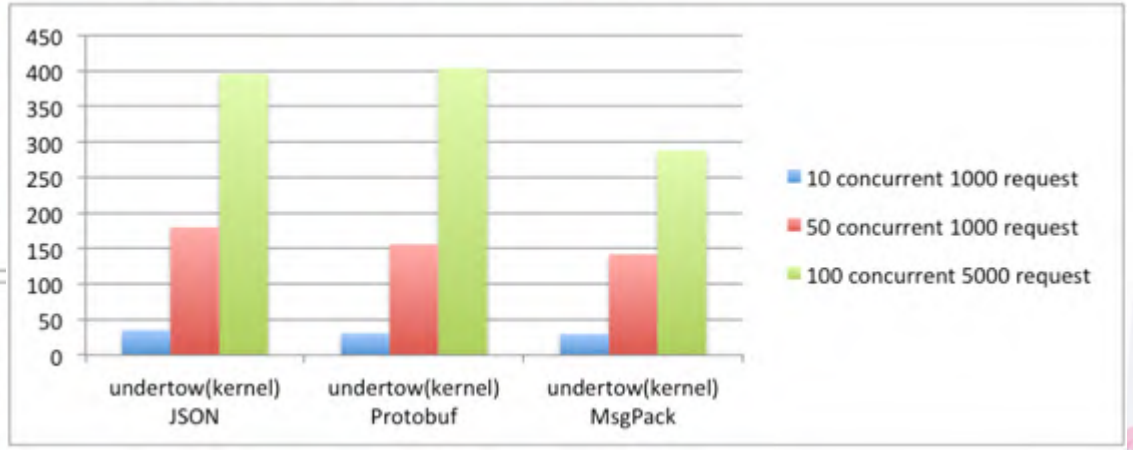
改造和优化

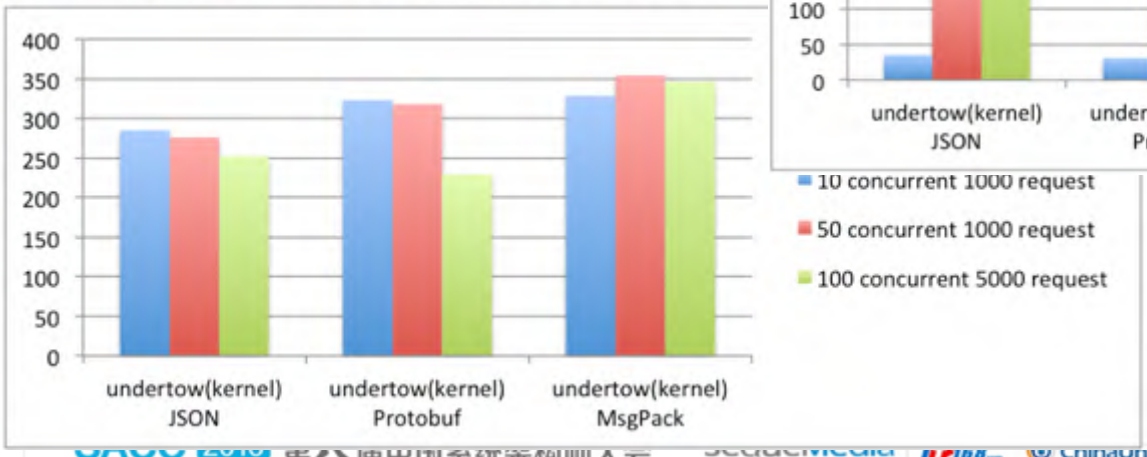

Neo4j Standalone Server
Unmanaged Extension
Kernel Extension
HTTP
NIO HTTP
Websocket
Jetty Server
JAX-RS
? Server
JSON
Protobuf
MsgPack

改造和优化


600
500
400
300
200
100
0
jetty(plugin) JSON
jetty(plugin) Protobuf
jetty(plugin) MsgPack
10 concurrent 1000 request
50 concurrent 1000 request
100 concurrent 5000 request

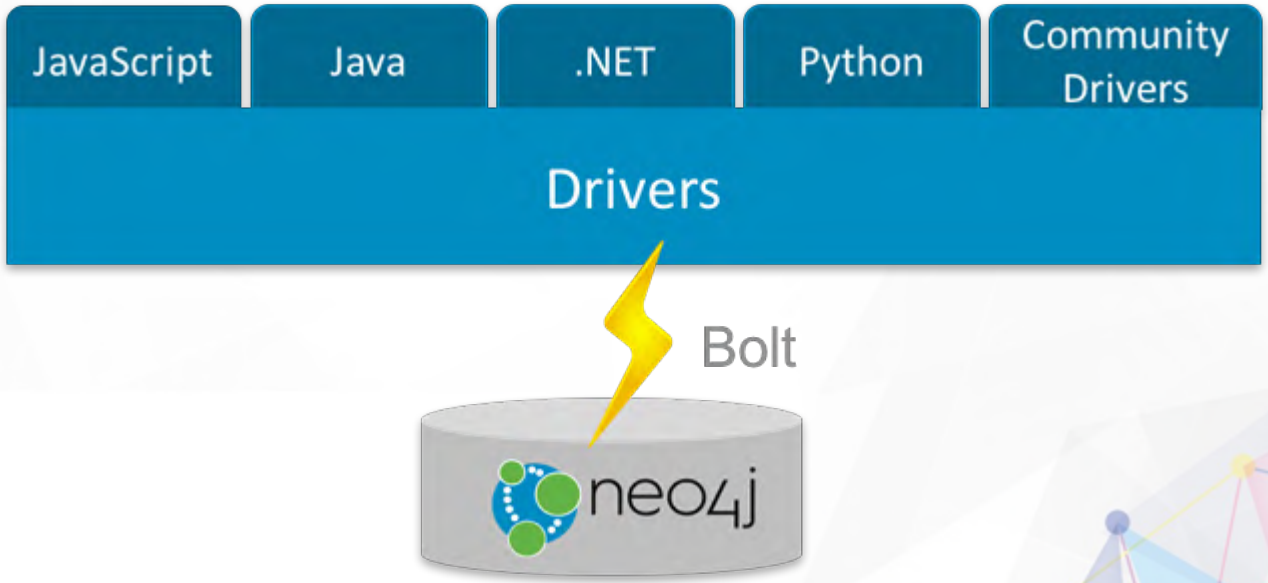

350
300
250
200
150
100
50
0
jetty(plugin) JSON
jetty(plugin) Protobuf
jetty(plugin) MsgPack


450
400
350
300
250
200
150
100
50
0
undertow(kernel) JSON
undertow(kernel) Protobuf
undertow(kernel) MsgPack
10 concurrent 1000 request
50 concurrent 1000 request
100 concurrent 5000 request


400
350
300
250
200
150
100
50
0
undertow(kernel) JSON
undertow(kernel) Protobuf
undertow(kernel) MsgPack
10 concurrent 1000 request
50 concurrent 1000 request
100 concurrent 5000 request

新特性

- Bolt：Streaming binary protocol
  - TCP persistent oriented
  - PackStream



新特性

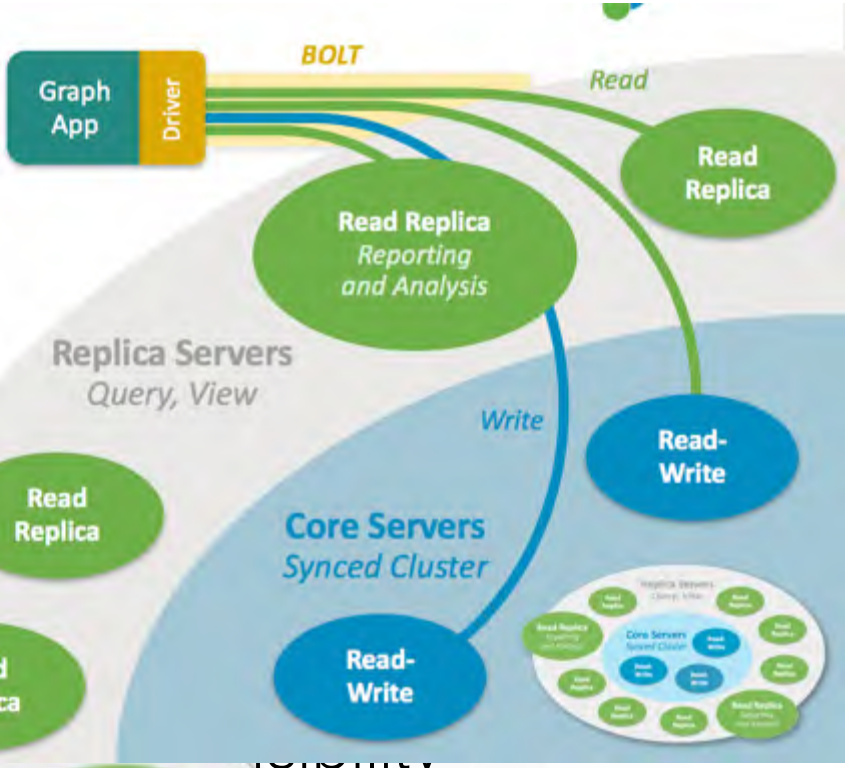
- Paxos->Raft
  - Paxos made ~~*moderately*~~ complex
  - more than “eventual consistency”, causal consistency

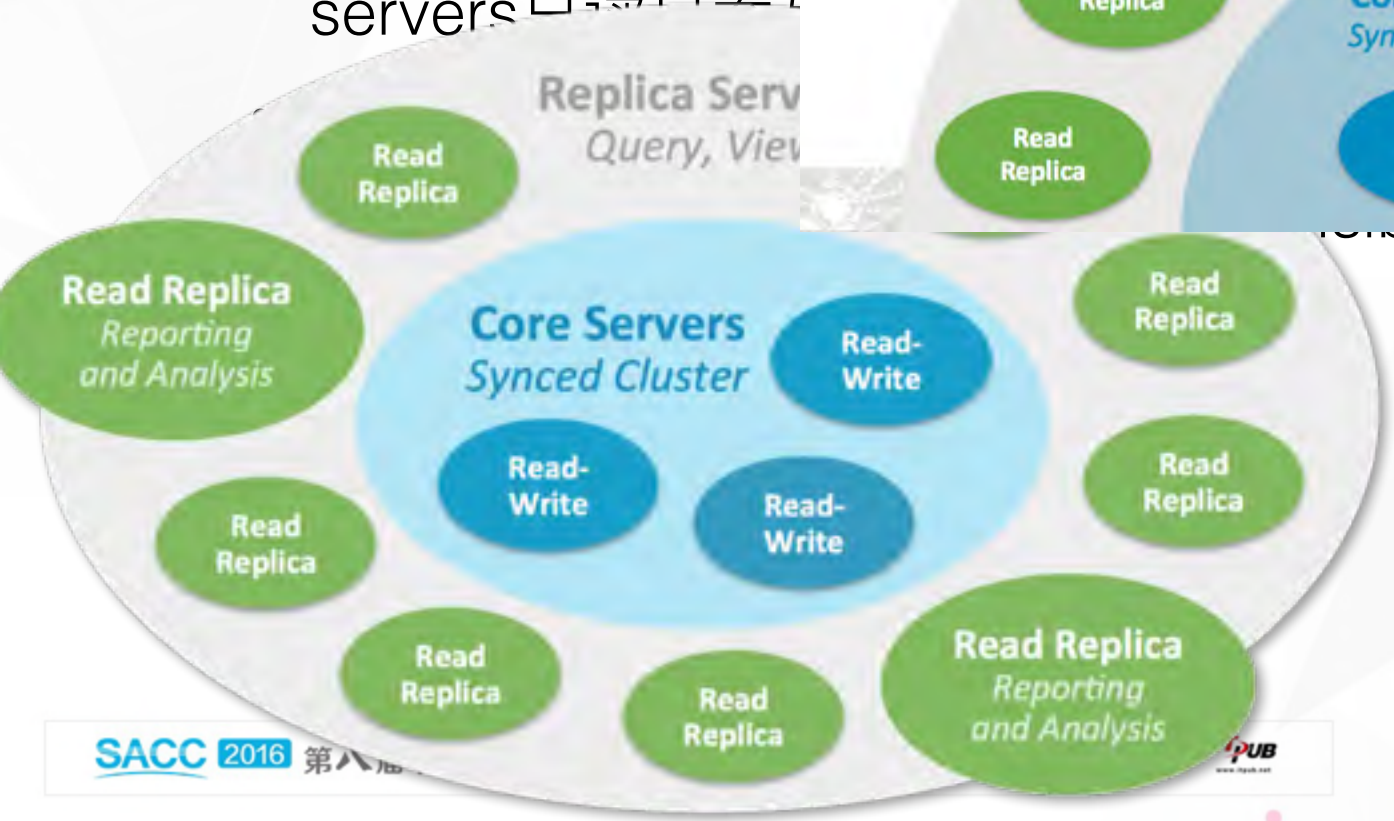
新特性

- New cluster
  - Core servers提供对等的多主结构，Replica servers只读只参与复制
  - Bolt+driver内置负载均衡的能力
  - Specify guarantees around query visibility
  - Stateful bookmark

新特性

- New cluster
  - Core servers提供
    servers只读只参与





新特性

- Security enhancement
  - role-based authorization
  - support LDAP/Active Directory
- Kernel/storage improvement
  - space compaction
  - automatic execution guard
- Administrative tools
- …

结束

Q & A

@Life_Player_

weijia_bj

THANKS

SequeMedia 盛拓传媒 | IT168 专注导购16年 | ChinaUnix | ITPUB