图数据库基准测试：

TigerGraph、Neo4j和Titan

测试报告摘要：作为全球首个原生并行图系统，TigerGraph™系统比 Neo4j 和 Titan 加载、存储和查询数据更快。

- 在使用单机时，TigerGraph 与 Neo4j 和 Titan 相比，**图遍历和查询响应时间快 4 倍至 100 倍**。
- 使用 TigerGraph，**查询速度几乎随服务器的数量线性增加**。即使在要求大量节点到节点通信的 PageRank 上，TigerGraph 仍然在 8 台服务器上实现了 4.7 倍的加速。
- 即使在使用单机时进行**在线加载数据**时，TigerGraph 固有的并行性使其能够**比 Neo4j 快 50 倍、比 Titan 快 25 倍**。
- TigerGraph 的高压缩比可以将**存储体积减少到 Neo4j 和 Titan 的 1/5 左右。**

本基准测试研究检测了 TigerGraph、Neo4j 和 Titan 的数据加载和查询性能。基准测试主题包括：

- 数据加载
  - 加载功能
  - 加载时间和速度
  - 被加载数据的存储体积
- 查询性能
  - 查询响应时间——针对 k 步查询
  - 查询响应时间——针对弱连通子图查询
  - 查询响应时间的可扩展性

1. 基准测试设置

测试的图数据库/分析系统版本：

- TigerGraph 0.8
- Neo4j 3.1.3 共享版
- TitanGraph 1.0.0

硬件平台

本次基准测试采用 Amazon EC2 实例类型。它适合测试大数据集，并且兼具强大的计算能力和经济性。

- 虚拟机：Amazon EC2，C3.8xlarge 实例类型
  - 32 个 vCPU，相当于 108 ECU
  - 60 GB 内存和 2x320 GB SSD 磁盘
- 附加存储：1TB Amazon EBS

表 1－数据集

名称	描述和来源	顶点	边
graph500-22	合成图 http://graph500.org	240 万	6400 万
twitter_rv.net	Twitter 用户—粉丝有向图 http://an.kaist.ac.kr/traces/WWW2010.html	4160 万	14.7 亿

对于每个图，原始数据被格式化为由单个制表符分隔的边流：

U1　　U3

U1　　U4

U2　　U3

顶点没有属性，因此不需要一个单独的顶点文件。

2. 数据加载测试

数据加载测试检测了以下三个方面：

- 加载功能
- 加载时间和速度
- 被加载数据的存储体积

加载功能

在考虑性能之前，我们首先比较功能性。这对了解每个图数据平台如何优化加载是必要的。

TigerGraph 以其原生 GSQL 语言编写加载任务，从而自动创建一个 RESTful 端点。然后可以通过 GSQL shell 或者通过向 RESTful 服务器提交 HTTP 请求来运行该任务。所有加载都在其他数据访问服务运行的同时在线发生。本测试采用 RESTful 接口的直接调用。

Neo4j 提供两个加载程序：一个使用 neo4j-import[1] 进行离线加载，另一个使用 LOAD CSV Cypher 命令在线加载。它们的功能有很大的差别。本次基准测试对这两种方法都进行了测试。

Titan 提供批量加载 [3] 的系统设置和建议，但不提供原始文件到数据库加载的语言。我们采用 Cassandrathrift 作为存储后端，并采用 Tinkerpop Graph Structure API 用 Java 编写了一个自定义加载程序。

---

1 https://neo4j.com/docs/operations-manual/current/tutorial/import-tool/

2 http://neo4j.com/docs/developer-manual/current/cypher/clauses/load-csv/

3 http://s3.thinkaurelius.eom/docs/titan/0.5.4/bulk-loading.html

表 2—加载功能对比

功能特点	TigerGraph	NEO4J-IMPORT	NEO4J-CYPHER	TITAN
内置加载语言	是	是	是	否
批量在线加载（1 亿到 1000 亿以上的边）	是	否	否	是
增量数据加载	是	否	是	是
加载过程中建立索引	是	否	是	是
顶点 ID 重复数据删除	是	否	是	否

4 √ TigerGraph 凭借批量在线加载、内置加载语言、内置索引等等，提供最完整而方便的加载环境。

5 加载时间和速度

为了在 Neo4j-import 中获得良好的性能，必须单独创建模式索引（schema index）。Neo4j 的总加载时间包括索引创建和数据加载。创建索引的时间显示在表 3 的括号中。随着加载进行，Neo4j Cypher 的加载速度越来越慢，因为它需要检查重复顶点的 ID。Neo4j Cypher 在 24 小时内没有完成加载测试，因此我们停止该加载，换为尝试加载到较小的数据集上。

表 3—数据集

名称	TigerGraph	NEO4J-OFFLINE	NEO4J-CYPHER	TITAN
graph500-22	139 秒	（13 秒）116 秒	24 小时内未完成	3002 秒
twitter_rv.net	1815 秒	（135 秒）3739 秒	24 小时内未完成	44554 秒

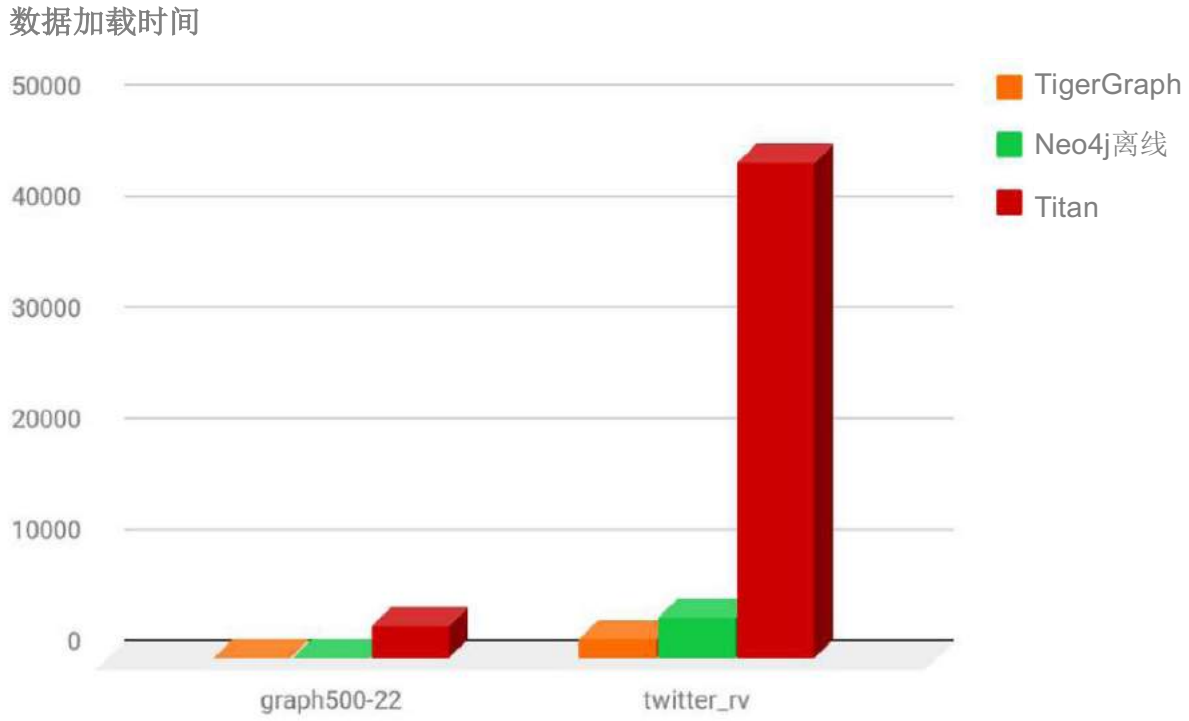

图 1—规范化数据加载时间

下面的图表显示了这些加载时间。

√ 在加载大数据集 twitter-rv 时，TigerGraph 比 Neo4j—offline 快两倍，比 Titan 快 25 倍。

通过将数据集的体积减少到 100 万行，Neo4j Cypher 的加载速度（每秒加载的数据行数）变得可测。在批量大小等于 10 万时，采用定期批量提交优化 Neo4j 的在线加载性能。表 4 包括了使用该较小数据集时 TigerGraph 和 Neo4j Cypher 的数据加载速度结果。

表 4—数据加载速度

数据集	TigerGraph	NE04J CYPHER
twitter-rv （前 100 万行）	809,000 行/秒	14,000 行/秒

下面的图表显示了加载时间。

平均加载速度，针对 14 亿的数据

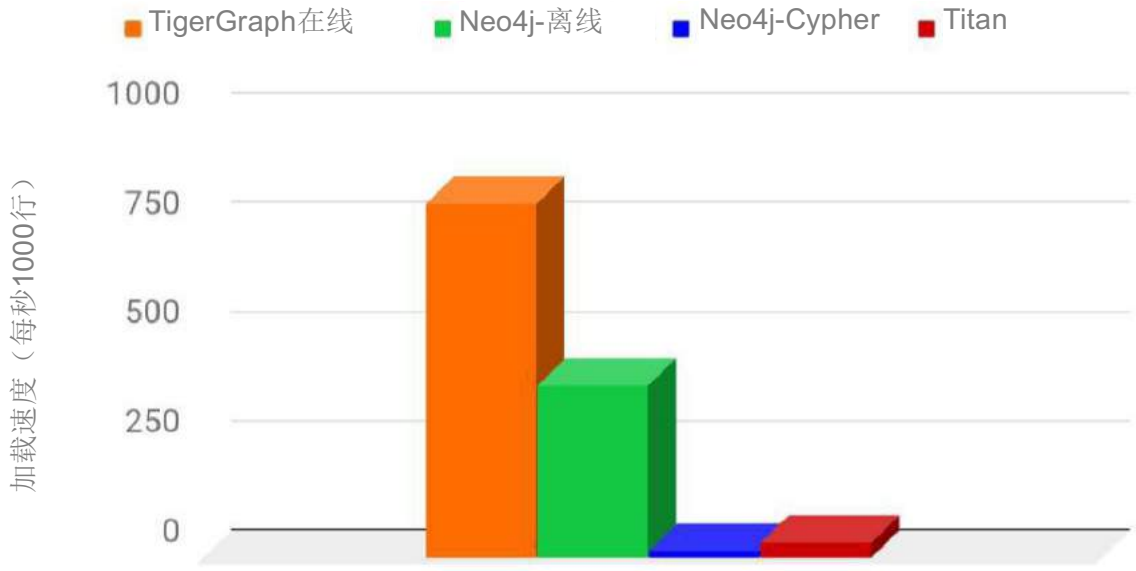

图 2—平均加载速度

√ **TigerGraph 加载 100 万条边比 Neo4j Cypher 快 50 倍以上。**

√ **Neo4j Cypher 甚至未能加载最小的数据集（6400 万条边）。**

被加载数据的存储体积

被加载数据落盘后的体积是重要的一个考量因素。TigerGraph 自动编码和压缩数据，将两个数据集的原始数据体积减少到原始体积的一半。相比之下，Neo4j 和 Titan 实际上将原来的数据体积增大了超过两倍。虽然两种 Neo4j 加载方法的操作方式不同，但得到的存储数据是相同的。

表 5—被加载数据落盘后的存储体积

数据集	原始数据	TigerGraph	NE04J	TITAN
graph500-22	967 MB	454 MB	2,369 MB	2,515 MB
twitter_rv.net	24,375 MB	11,038 MB	51,047 MB	50,028 MB

√ **为存储这些大数据集，TigerGraph 使用的空间大约为 Neo4j 和 Titan 所使用空间的 1/5 左右。**

标准化数据库体积

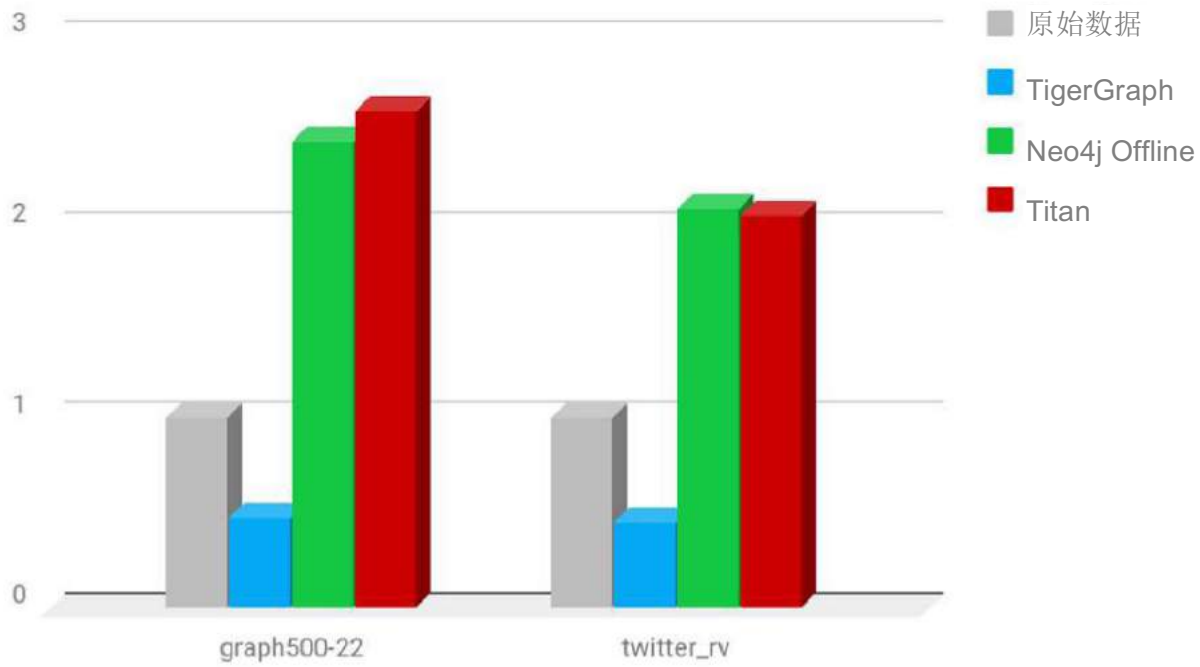

图 3—加载后标准化数据库体积

数据加载小结

√ 只有 TigerGraph 能够对大规模数据进行在线数据加载。

√ TigerGraph 的高压缩比可以将存储体积减少到 Neo4j 和 Titan 的 1/5 左右。

3.查询性能测试

查询性能测试检测了以下三个方面：

- 查询响应时间——针对 k 步
- 查询响应时间——针对弱连通子图
- 查询响应时间的可扩展性

查询响应时间—针对k步

k-步邻居查询是衡量图遍历性能的一个很好的标准，它询问在起始顶点的 k 步的所有顶点。一个顶点的一步邻居就是其 • 相邻顶点的集合。我们在每个数据集中随机选择 1000 个起始顶点，计算平均查询时间。每个起始顶点具有不同的邻居数量 N；我们报告 N 的平均值。我们从客户端测量查询执行时间（客户端在与数据库相同的服务器上运行），其中包括网络传输时间。为了集中比较图遍历和最小化网络输出时间，我们仅输出 k 步邻居集合的大小，而不是完整的邻居列表。

TigerGraph 和 Neo4j 都能有效地实施 k 步邻居查询。TigerGraph 允许用户调整线程数量。在这些测试中，使用了 16 个线程。Titan 使用了 Gremlin 查询语言。我们发现其实施 k 步查询的唯一合理方法是使用子图 API。测试显示，子图 API 非常耗费资源，使得两步邻居查询很难完成。相反，当使用离源头刚好 k 个步长的顶点集计算 k 步邻居时，Titan 表现得更好。下表包括了 TigerGraph、Neo4j 和 Titan 使用 k-步以及 Titan 使用 k-shell 近似的查询时间。

表 6——一步邻居查询时间

数据集	AVG N	TigerGraph	NEO4J	TITAN（k-步邻居）	TITAN（k-SHELL 近似）
graph500-22	4082	3.3 毫秒	14.4 毫秒	3064.9 毫秒	14.2 毫秒
twitter_rv.net	12271	7.5 毫秒	55.0 毫秒	34.1 毫秒	39.8 毫秒

表 7—两步域查询时间

数据集	AVG N	TigerGraph	NE04J	TITAN（k-步邻居）	TITAN（k-SHELL 近似）
graph500-22	457400	0.19 秒	19.0 秒	内存不足	11.1 秒
twitter_rv.net	1747130	0.77 秒	21.2 秒	61.6 秒	58.7 秒

下表省略了 Titan 对于 Graph500-22 数据集的一步邻居查询时间。响应时间值如此之高（3064.9 毫秒），远远超出了其他值的范围，这将使图表无法读取。

一步邻居查询时间（毫秒）

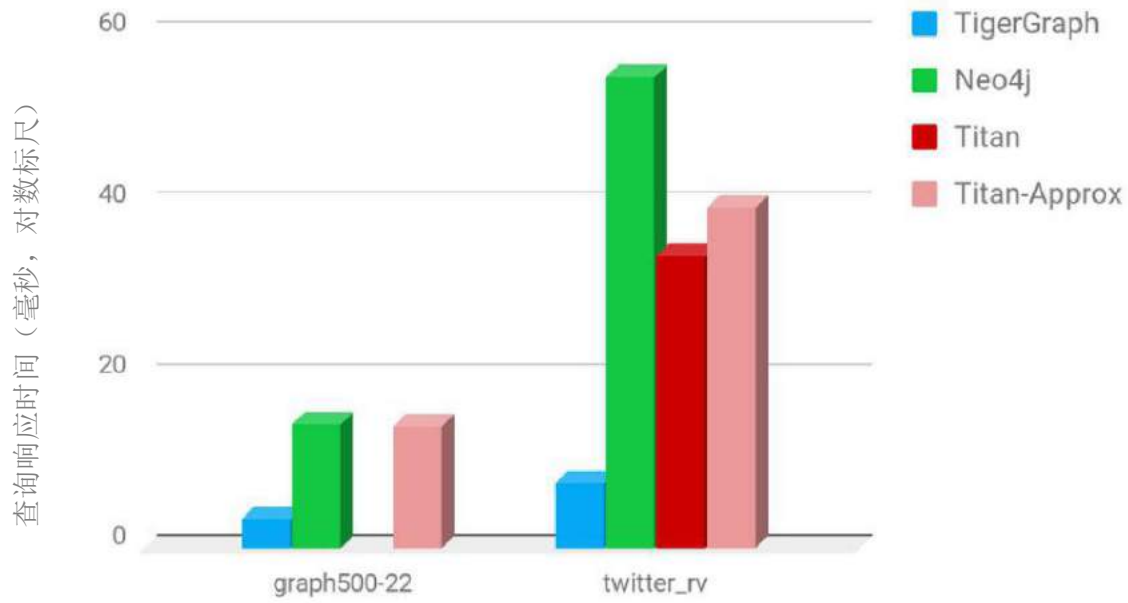

图 4—一步邻居查询时间

在两步邻居情况中，Titan 对于 graph500-22 数据集内存不足。

两步邻居查询时间（秒）

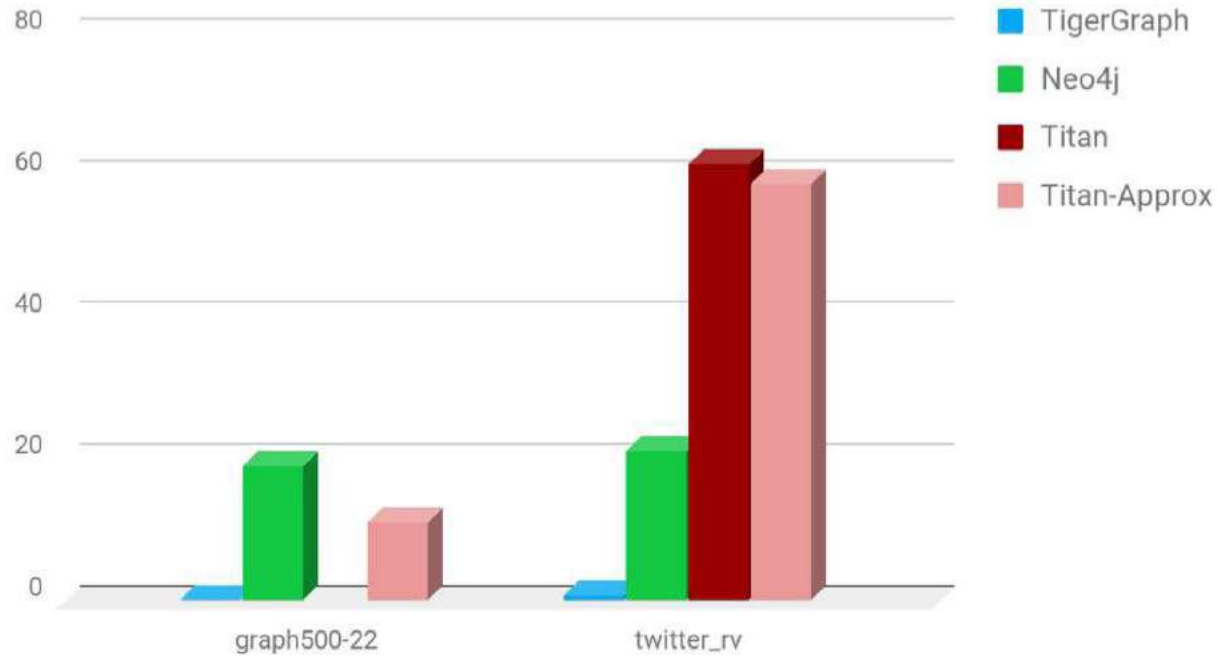

图 5—标准化两步邻居查询时间

√ **TigerGraph 在一步邻居查询上比 Neo4j 快 4 倍至 7 倍，并且在两步邻居查询上比 Neo4j 快 27 倍至 1OO 倍。**

√ **Titan 在一步邻居查询上比 TigerGraph 慢 900 倍，不能完成两步邻居查询。**

查询响应时间—针对弱连通子图

如果忽略有向图的方向，则弱连通子图（WCC）是可以相互接触的顶点及其连接边的最大集合。弱连通子图查询查找图中的所有弱连通子图（WCC）。该查询要求遍历每个顶点和每条边。

我们将 TigerGraph 与 Neo4j 进行了比较。TigerGraph 的 GSQL 查询语言足够灵活，使我们得以尝试多种方法。我们用两种方式实施弱连通子图（WCC）。第一，WCC-global 使每个顶点成为一个起点，并从每个顶点扩展，以并行找到弱连通子图（WCC）；第二，WCC-BFS 从一个未标记的顶点运行宽度优先的搜索算法，为其分配一个新的弱连通子图 ID，并使用相同的弱连通子图 ID 标记所有可达到的顶点。WCC-global 具有更好的并行性，而 WCC-BFS 算法复杂度较低。TigerGraph 的 WCC-BFS 比 WCC-global 大约快了两倍，所以我们只报告了 WCC-BFS 时间。

对于 Neo4j，我们采用了 Neo4j AOPC 程序（3.1.3.6 版）中提供的 Neo4j 原生弱连通子图实现。APOC 的 WCC 实现采用类似于 WCC-BFS 的算法。

为了最小化网络开销时间，我们设计了查询，使之仅返回弱连接组件的数量，而不是报告每个组件的完整成员集。

表 8－弱连通子图查询时间

数据集	TIGERGRAPH-BFS-16	NE04J
graph500-22	8.7 秒	73.0 秒
twitter-rv	217.8 秒	内存不足

√ 在较小的数据集上，TigerGraph 比 Neo4j 快 8 倍。

√ 在较大的 twitter-rv 数据集上，Neo4j 的内存占用更大，并且在运行时内存不足。

查询响应时间的可扩展性

在我们的最终测试中，我们检测 TigerGraph 的原生并行图架构，探究它如何使查询响应时间随着使用的服务器数量增加而减少。对于该测试，我们使用了略微不同的 Amazon EC2 实例类型（DB.R3.2xlarge）。我们使用了 twitter_rv 数据集，并且对有名的 PageRank 查询（每个顶点打分）重复了 10 次。我们并行测试了 1、2、4 和 8 台服务器。图在所有使用的服务器上被分割成大小相同的片段。

表 9－查询响应时间的可扩展性

服务器数量	1	2	3	4
平均查询时间	81.4 秒	51.8 秒	28.9 秒	17.2 秒
加速	1.0	1.57	2.81	4.73

PageRank 是一种迭代算法，它在每次迭代期间遍历每条边。这意味着服务器之间有很多通信，信息在不同分区之间相互发送。尽管存在这种通信开销，TigerGraph 的原生并行图架构仍然在 8 台服务器上成功地实现了 4.7 倍的加速。

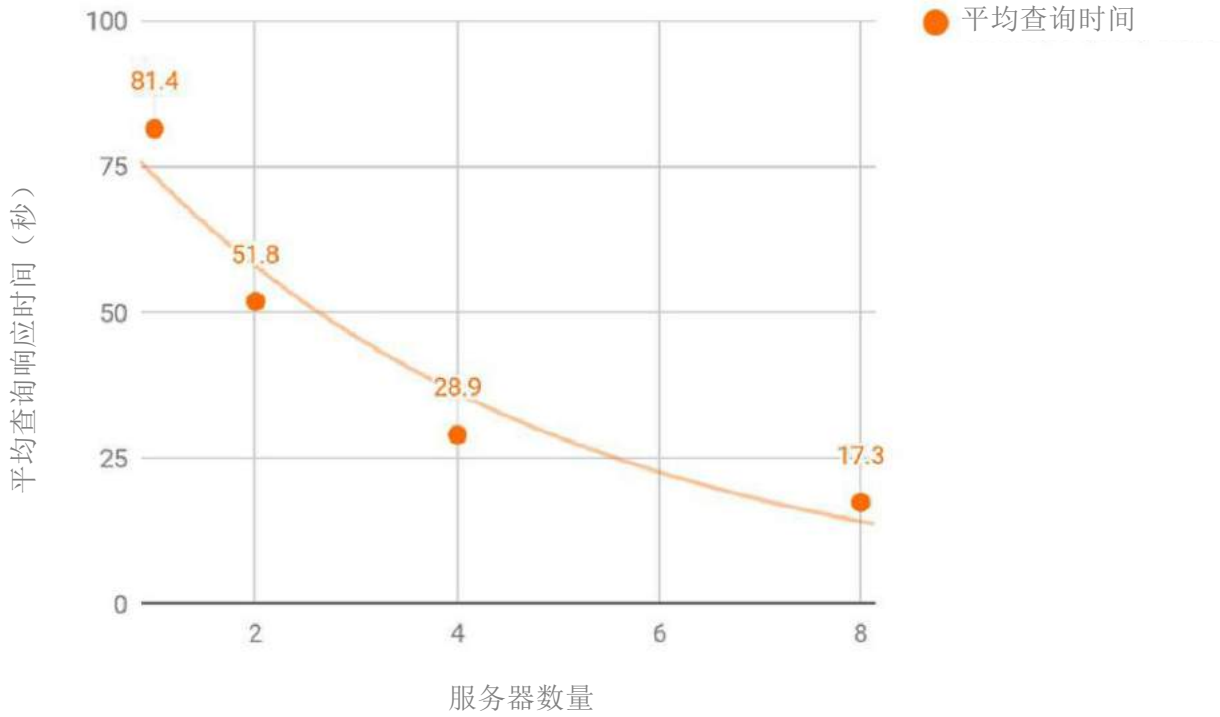

图 6－查询响应时间与机器数量

Neo4j 是单个服务器，而不是分布式系统。因此，Neo4j 不能将一个图分割到多台服务器上，也不能处理大图数据。TigerGraph 是首个和唯一的真正分布式原生并行图系统。TigerGraph 中的图可以分割并存储在多台服务器上。

查询性能小结

√ 与 Neo4j 和 Titan 相比，TigerGraph 的图遍历和查询响应时间快 4 倍至 100 倍。

√ TigerGraph 的查询速度随着服务器数量增加而提高，从而对 8 台服务器实现了 4.7 倍的速度递增。

关于TigerGraph

TigerGraph是基于原生并行图（NPG）技术的全球首个实时图分析平台。TigerGraph通过为具有复杂和海量数据的企业提供实时深度链接分析支持，实现图平台的真正承诺和好处。TigerGraph的成熟技术已经被支付宝、VISA、软银、中国国家电网公司、Wish、Elementum等客户所采用。

TigerGraph由许昱博士于2012年创立，并获得启明创投、百度、蚂蚁金融、AME云创投、莫拉多风险投资公司、佐德·纳齐姆、丹华资本和DCVC风投基金公司的资助。TigerGraph的总部位于加利福尼亚州红木市。如欲了解更多信息，请访问www.tigergraph.com。