

Как мы храним большой социальный граф

Бартенев Максим
Норси-Транс



Конференция разработчиков
высоконагруженных систем



План доклада

- Что мы решали с помощью графовых БД
- Графовые БД Neo4J и Sparksee
- Настройка и оптимизация Neo4J и Sparksee
- Каких результатов удалось достичь

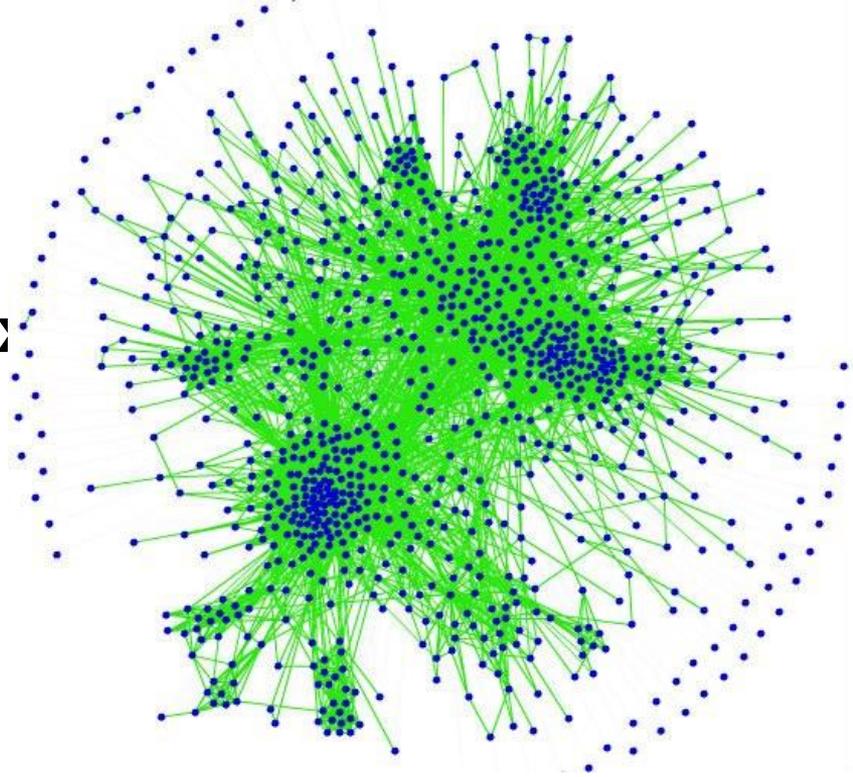
Графы везде

Применяются во многих сферах:

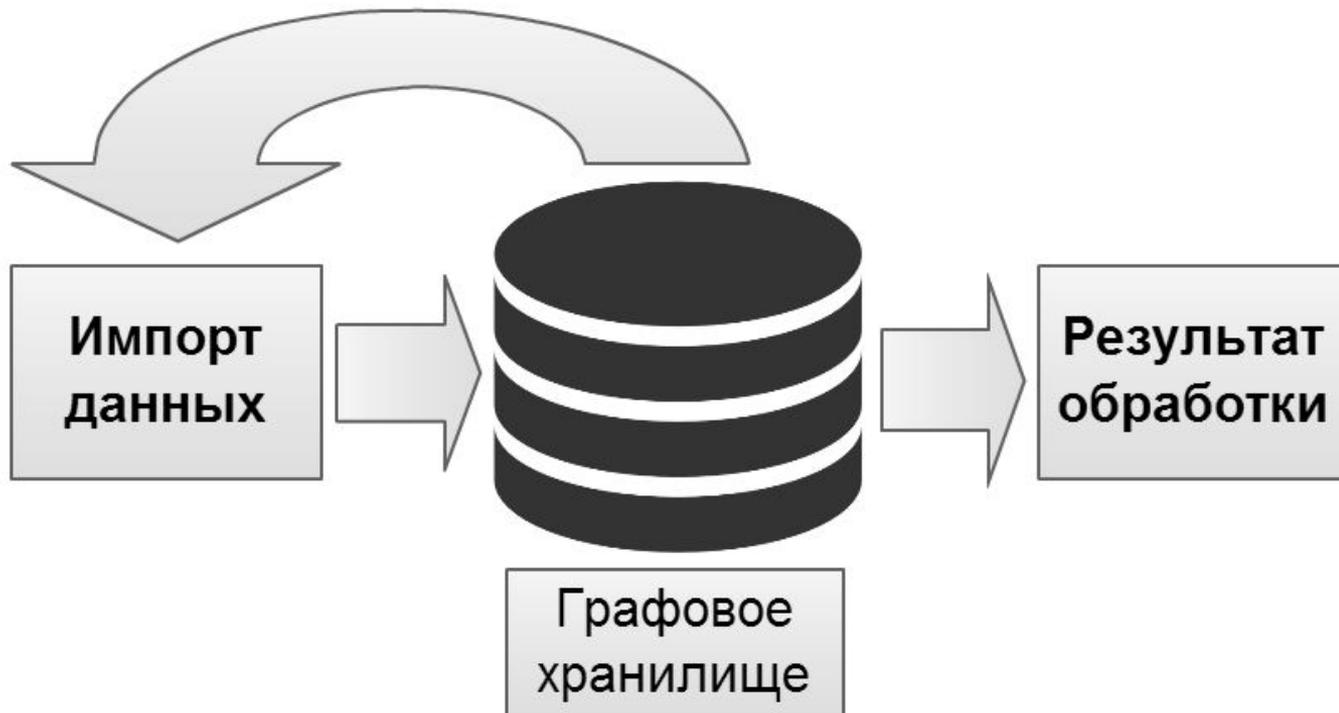
- веб-ссылки;
- маршруты;
- социальные сети;
- и т.д.

Имеют очень большой объем.

Сложность в анализе графа, а не в хранении.



Графовое хранилище



Решаемые задачи

- Загрузка графа
- Выполнение аналитической операции
- Догрузка новых данных, в случае их появления

Аналитические задачи

- Получить всех соседей вершины (Neighbors)
- Выполнить обход графа (BFS)
- Найти кратчайший путь (Shortest path)

Neo4J

- Наиболее распространенная
- Развитое сообщество
- Высокая функциональность
- Может быть как серверным приложением, так и встраиваемым
- Есть бесплатная версия



Особенности Neo4J

- Все операции только внутри транзакции – правильно и надежно, но медленно и ест много оперативной памяти.
- Объекты – вершины, ребра и атрибуты. Доступ к ним только по внутреннему идентификатору.



BatchInserter

- Быстрый импорт
- НЕ отказоустойчивый
- НЕ потокобезопасный



Индексирование

- Новый метод `schema.indexFor()` – только по атрибутам на вершинах
- Устаревший метод `graphDb.index()` – и по вершинам и по ребрам
- Индексация в режиме Batch inserter
`BatchInserterIndexProvider.nodeIndex()`

Memory mapped cache

- Служит для ускорения I/O
- Проецирует файлы хранилища в память
- Каждому файлу свой кэш

Размеры объектов на диске

Вершина 15 В

Ребро 34 В

Атрибут 41 В

Строка 128 В

Массив 128 В

Cache size = размер объекта * количество объектов



Настройки memory mapped cache

- use_memory_mapped_buffers
- mapped_memory
 - nodestore.db.mapped_memory
 - relationshipstore.db.mapped_memory
 - propertystore.db.mapped_memory
 - И Т.Д.



Object cache

- Хранит в себе объекты для быстрого доступа при обходах графа
- Вытеснение объектов осуществляет GC
- Реально производительный кэш есть только в Enterprise версии

Типы Object cache

none кэш отсутствует

soft стандартное значение в бесплатной версии

weak больше данных при меньшем времени их жизни

strong хранит в себе все объекты графа

hpc специальный высокопроизводительный кэш



Sparksee (в прошлом DEX)

- Заявлена высокая производительность
- Только встраиваемая
- Не столь распространенная
- Сообщество очень маленькое
- Полностью закрытая
- Бесплатна для исследований



Особенности Sparksee

- Обязательно задается схема данных
- Доступ к объекту только по внутреннему идентификатору

Настройки Sparksee

- Настройки ребер:
 - Ориентированные
 - Индексированные
- Типы атрибутов:
 - Обычный
 - Индексированный
 - Уникальный

Sparksee cache

- Настройки кэширования минимальны
- Все новые объекты попадают в кэш
- `SetCacheMaxSize(int megabytes)`
 - Если `megabytes == 0`, то используется вся свободная память минус 512mb.

Тестовый стенд

- Intel Xeon E7540 2.0 GHz
- 64GB DDR3
- 2x2TB hard drive

ПО и настройки Neo4J

- Neo4J 2.1.5 Community Edition
- Ubuntu 14.04 LTS
- JVM: -d64 -Xmx40G -XX:+UseParallelGC
- Batch insertion mode
- Use_memory_mapped_buffers
- Cache vertices 2GB, relationships 18GB

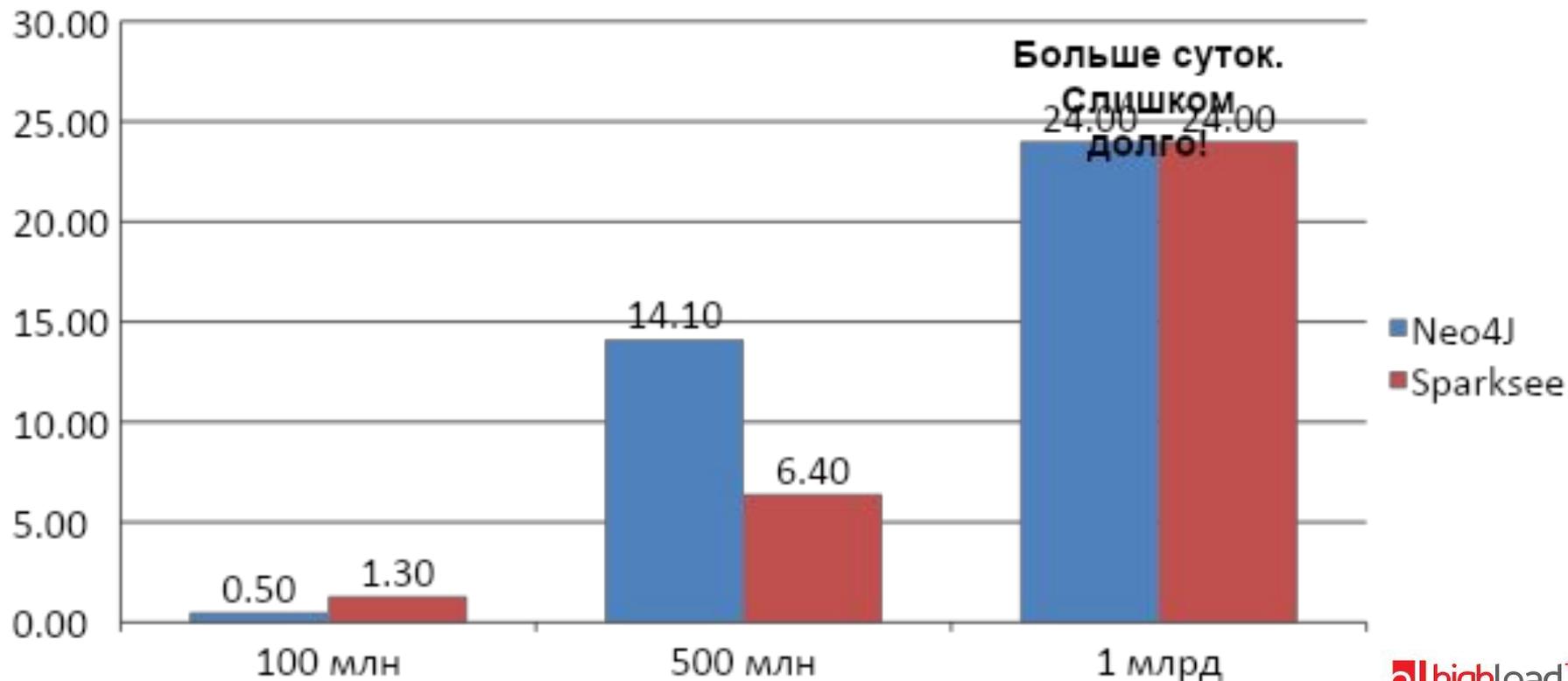


ПО и настройки Sparksee

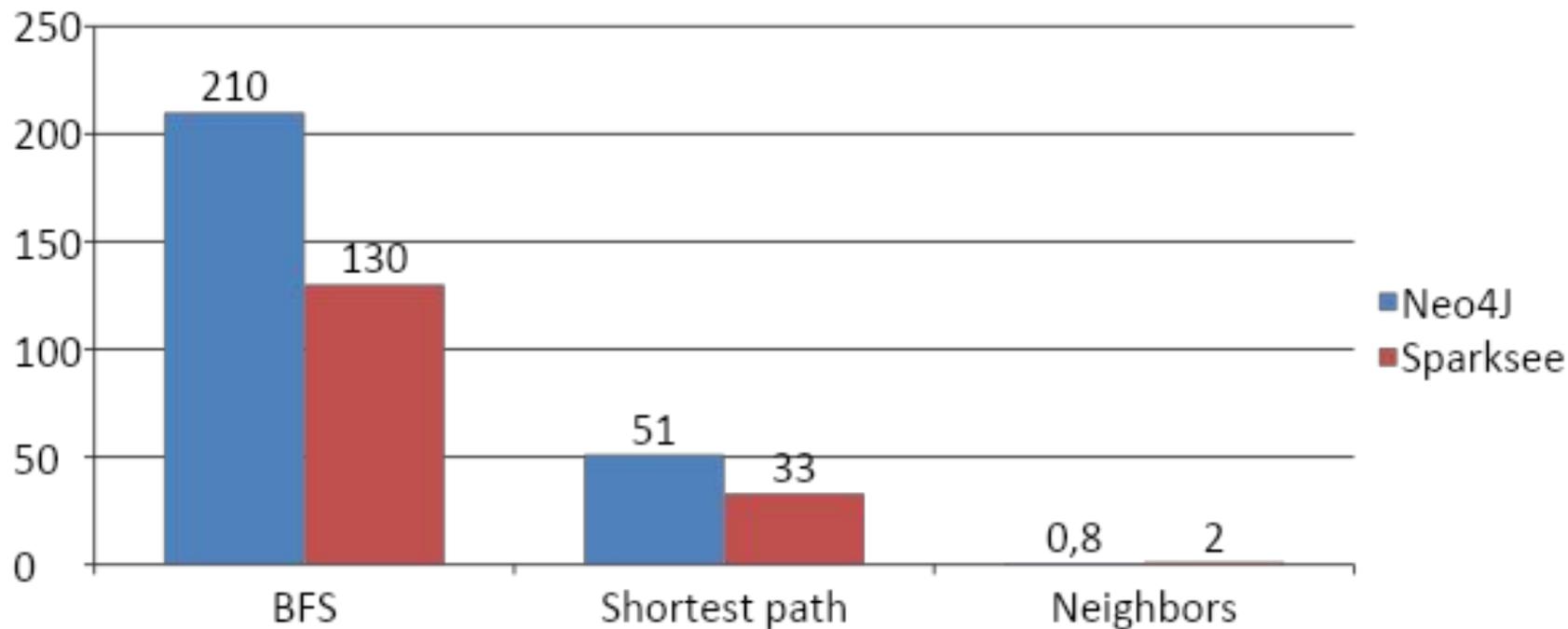
- Sparksee 5.1.0 Unlimited licence
- Windows Server 2008 x64
- .NET API
- Cache size 60GB



Время импорта данных (ч)



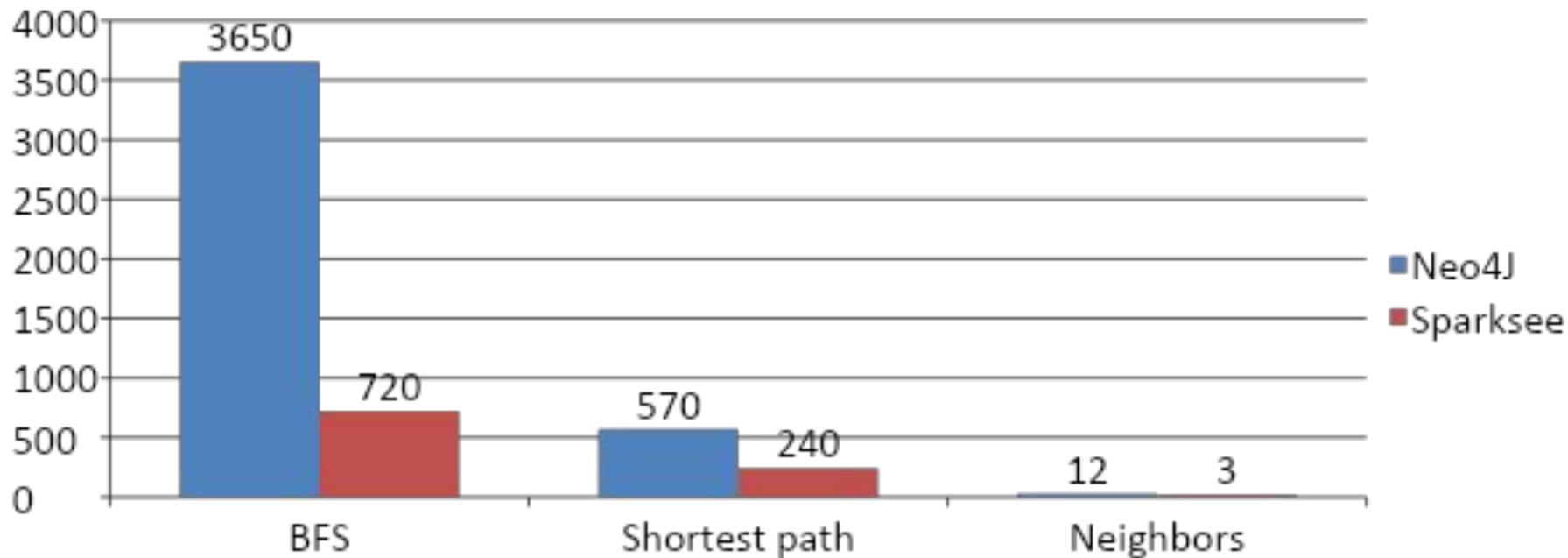
Время обработки графа (с)



~10
МИЛЛ



Время обработки графа (с)



~50
МИЛЛ



Выводы

- Sparksee производительнее Neo4J
- Высокая производительность графовых БД ограничивается размером памяти
- Графы размером больше 1 млрд вершин не получится обработать



**Спасибо
за
внимание!**

