



5 Ways to Accelerate and Scale Out PostgreSQL

Denis Magda

Apache Ignite Committer and PMC Chair

GridGain VP of Product Management

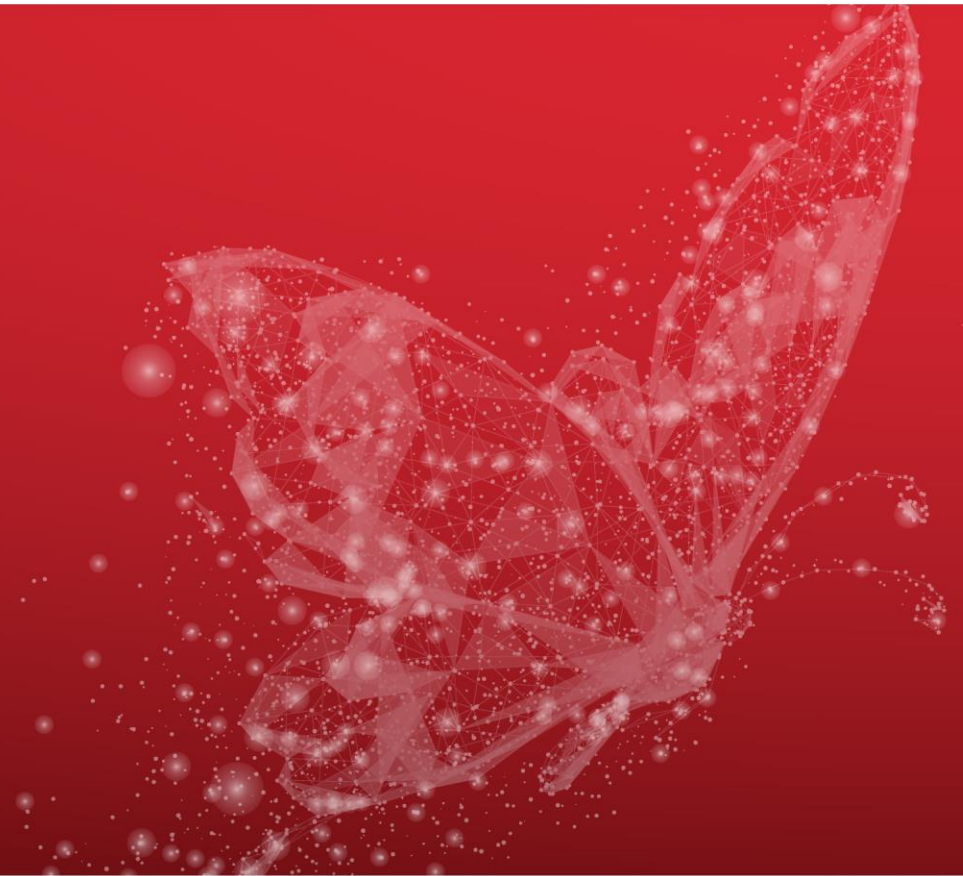


Agenda



- Tapping into RAM with caching techniques
- Sharding and replication solutions
- Cache and scale out with in-memory data grids
- Q&A

Caching Techniques



Ultimate Purpose of Caching



Speed up operations by reducing
disk access and **computation** (i.e. CPU)

Computer Latency at Human Scale



System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	1 s
Level 1 cache access	0.9 ns	2 s
Level 2 cache access	2.8 ns	7 s
Level 3 cache access	28 ns	1 min
Main memory access (DDR DIMM)	~100 ns	4 min
Intel Optane DC persistent memory access	~350 ns	15 min
Intel Optane DC SSD I/O	< 10 μ s	7 hrs
NVMe SSD I/O	~25 μ s	17 hrs
SSD I/O	50-150 μ s	1.5 - 4 days
Rotational disk I/O	1 – 10ms	1 – 9 months
Internet: SF to NY	65 ms	5 years

Computer Latency at Human Scale

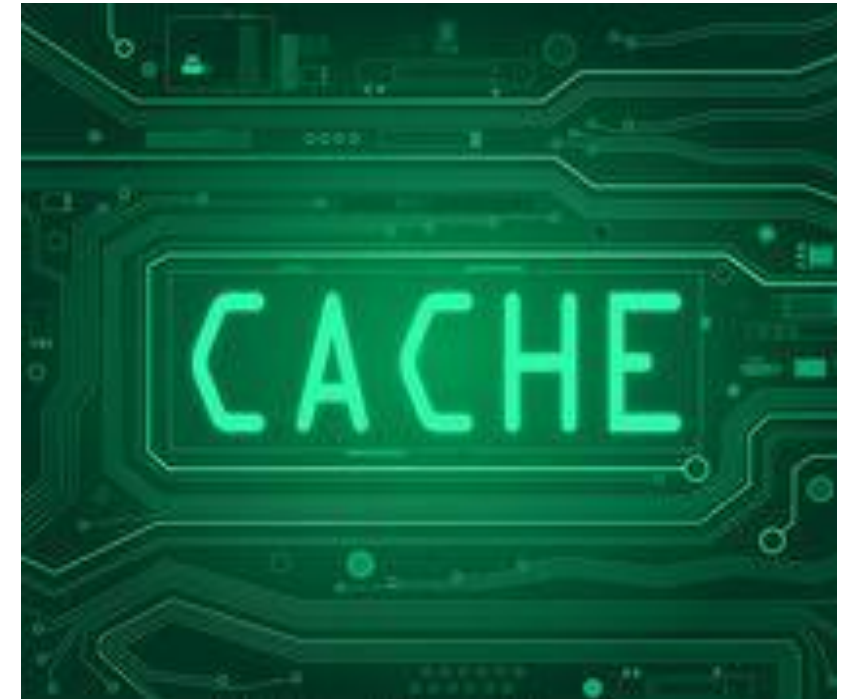


System Event	Actual Latency	Scaled Latency
One CPU cycle	0.4 ns	1 s
Level 1 cache access	0.9 ns	2 s
Level 2 cache access	2.8 ns	7 s
Level 3 cache access	28 ns	1 min
Main memory access (DDR DIMM)	~100 ns	4 min
Intel Optane DC persistent memory access	~350 ns	15 min
Intel Optane DC SSD I/O	< 10 μ s	7 hrs
NVMe SSD I/O	~25 μ s	17 hrs
SSD I/O	50-150 μ s	1.5 - 4 days
Rotational disk I/O	1 – 10ms	1 – 9 months
Internet: SF to NY	65 ms	5 years

Basic Types of Caching in Postgres



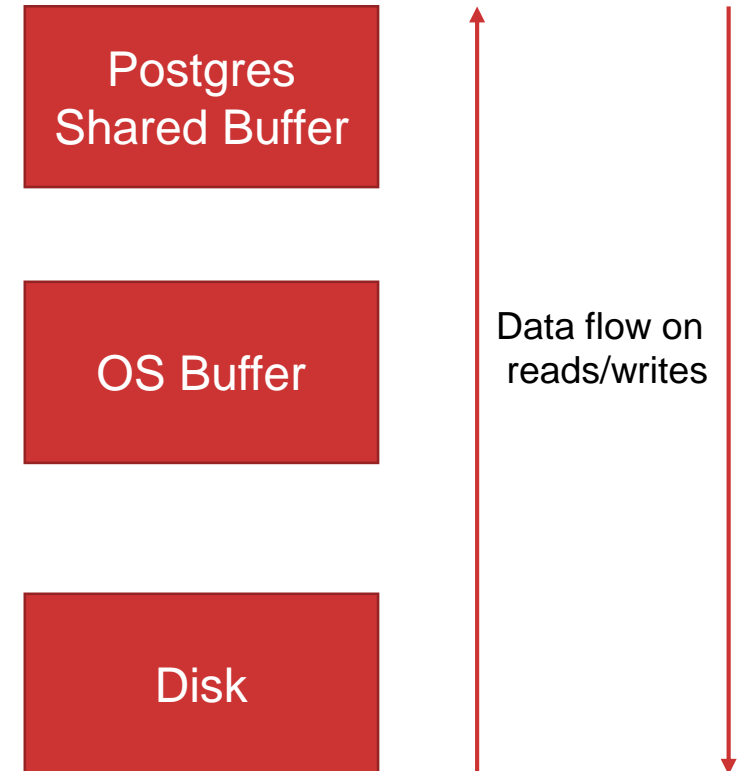
- Query result caching
- Query plan caching
- Relation caching
 - Data and indexes



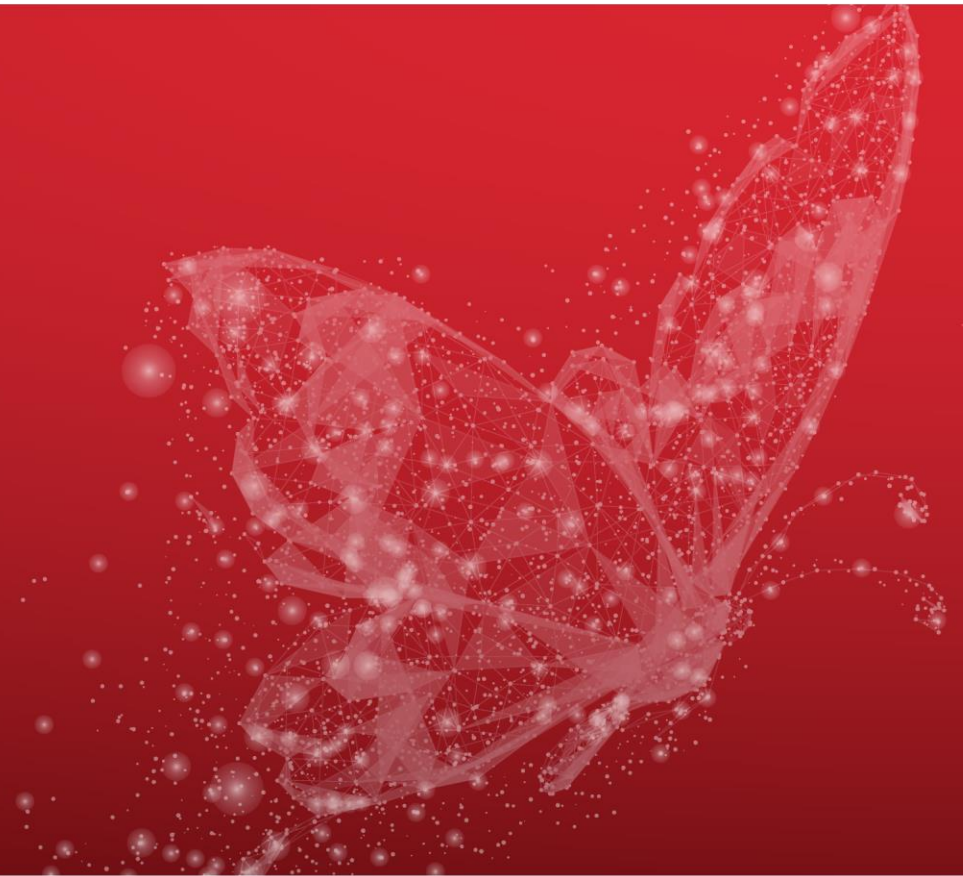
Relation caching: Shared Buffer and OS Buffer



- Postgres Shared Buffer Cache
 - Allocated and managed by Postgres
- OS Buffer (aka. Page Cache)
 - Caches chunks (pages) of files
- Suggestions/considerations:
 - No silver bullet – select and tune
 - Possible duplication between shared and OS caches
 - Limited by local RAM capacity



Horizontal Scalability



Defining Requirements for Solution

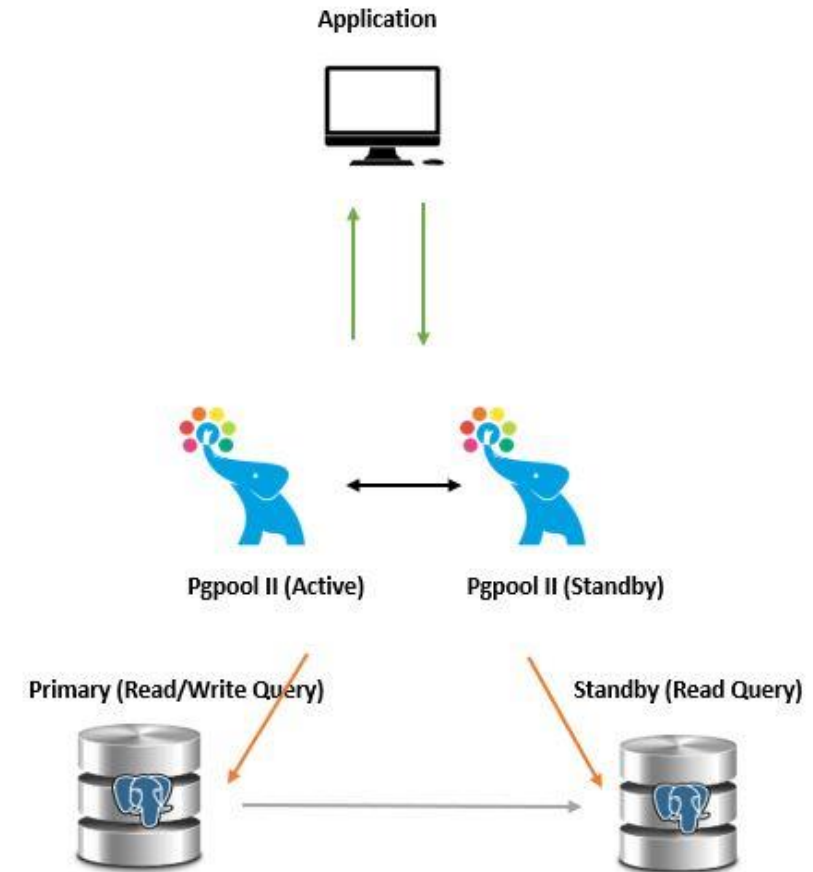


- Strong Consistency (ACID)
- Load Balancing
- High-Availability and Failover

Pgpool 2 for Read-Heavy Workloads



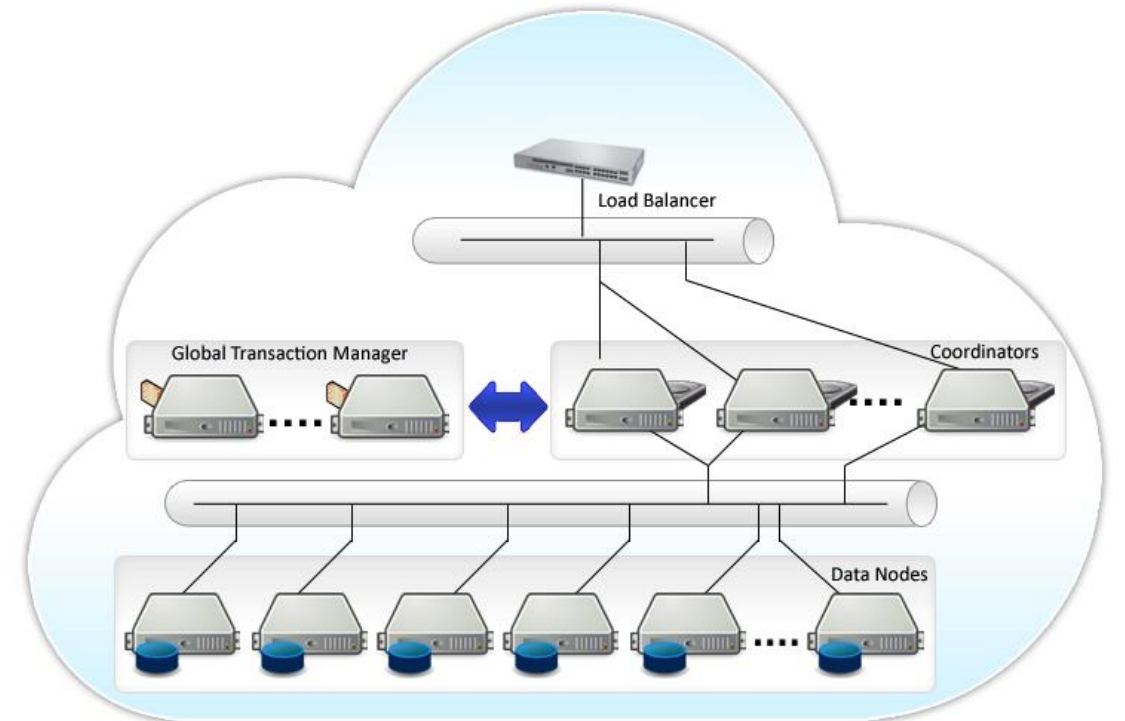
- Pgpool coordinator
- Primary for reads and writes
- Hot replicas for reads
- Suggestions/considerations:
 - Good for load balancing of read-heavy workloads
 - ACID enforces sync replication and limits a number of replicas
 - Primary machine capacity defines your total cluster capacity



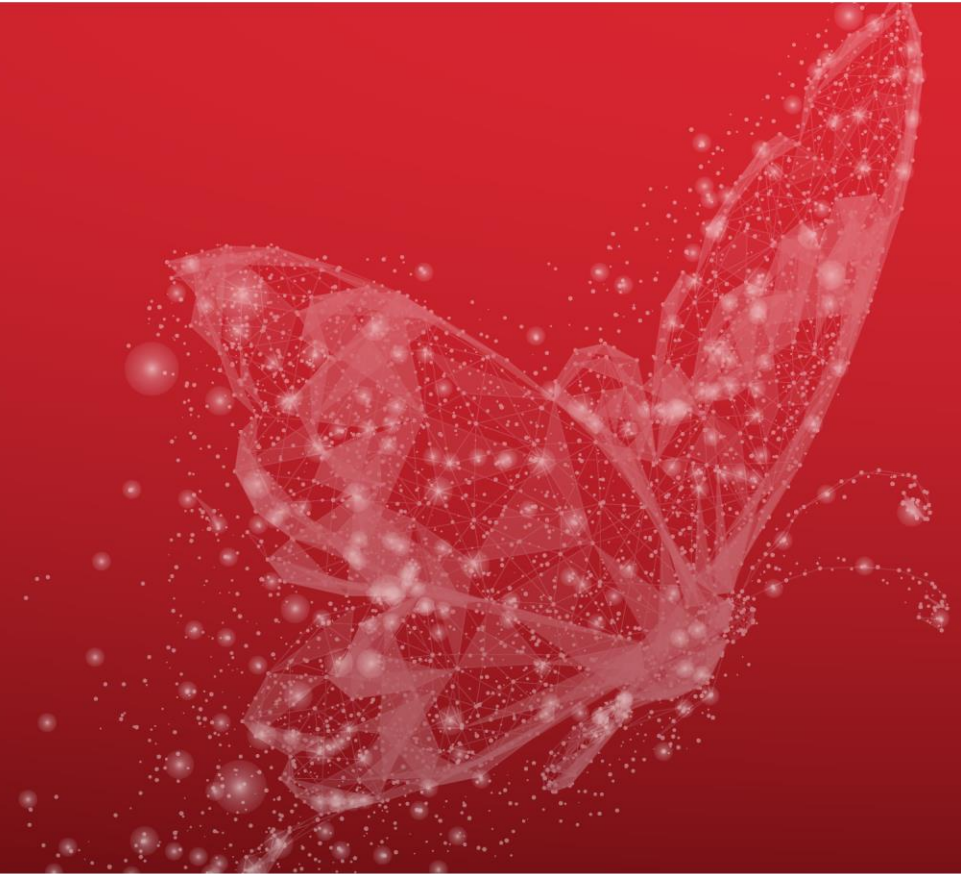
Sharding With PostgreSQL-XL or CitusData



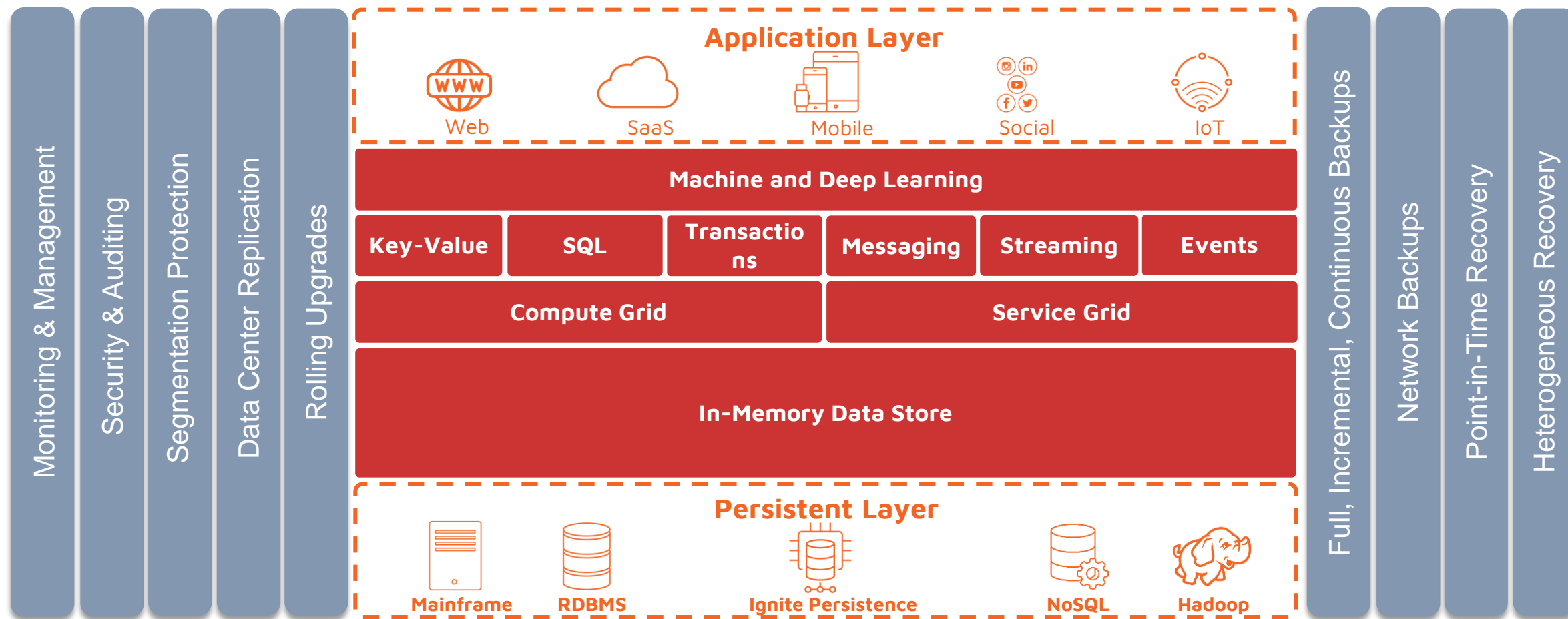
- Coordinator keeps metadata and distributes queries
- Data nodes store shards/partitions
- Supports data co-location and JOINS
- Suggestions/considerations:
 - Suited for mixed workloads
 - Total capacity is your cluster capacity
 - Scaling and failover is not trivial
 - Disk-based solution



Caching and Scaling With In-Memory Data Grids



Apache Ignite/GridGain In-Memory Computing Platform

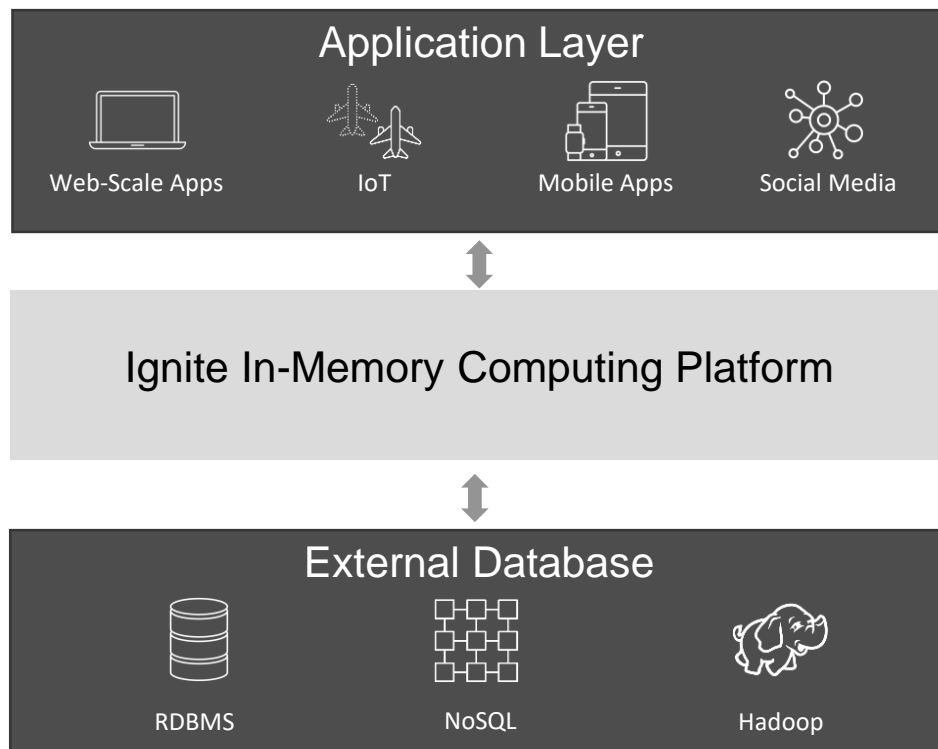


Apache Ignite Features
GridGain Enterprise Features

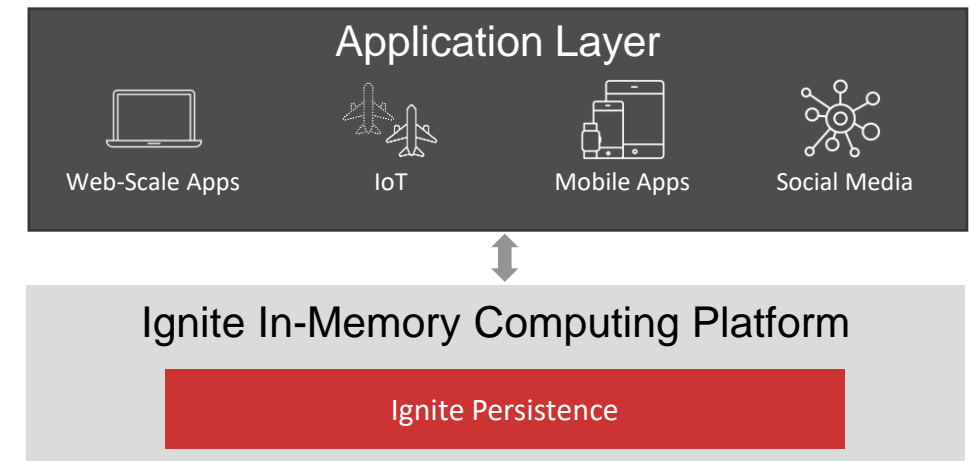
Primary Ignite Deployment Modes



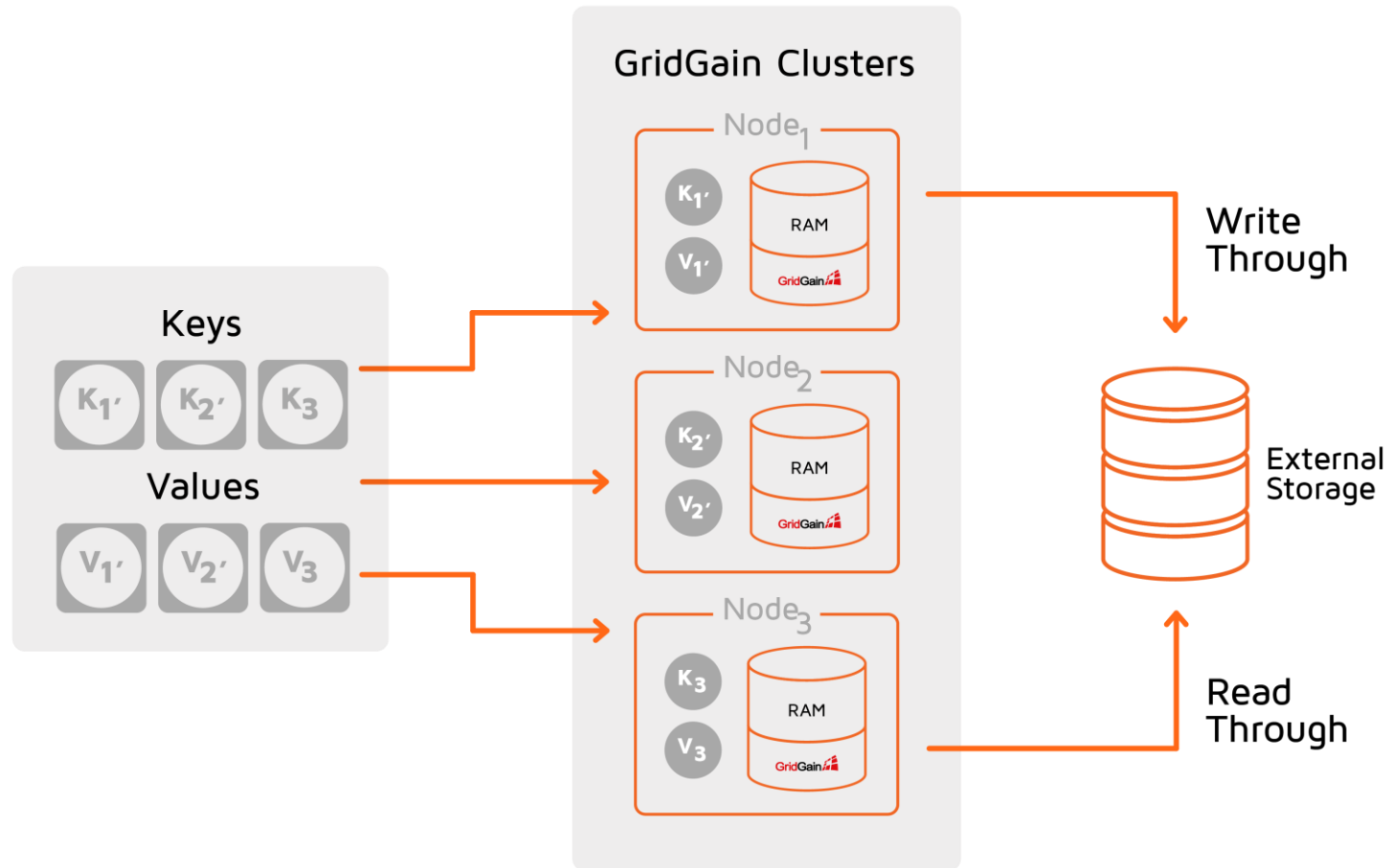
Enhance Legacy Architecture - IMDG



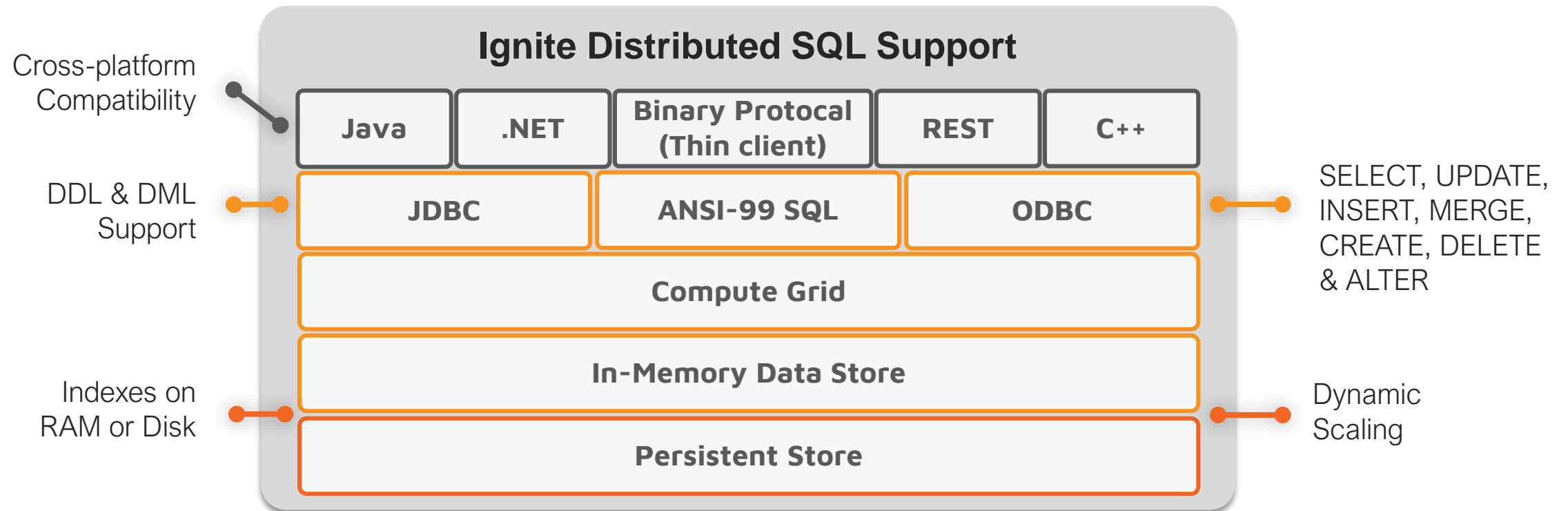
Simplified Modern Architecture - IMDB



How Postgres is Accelerated?

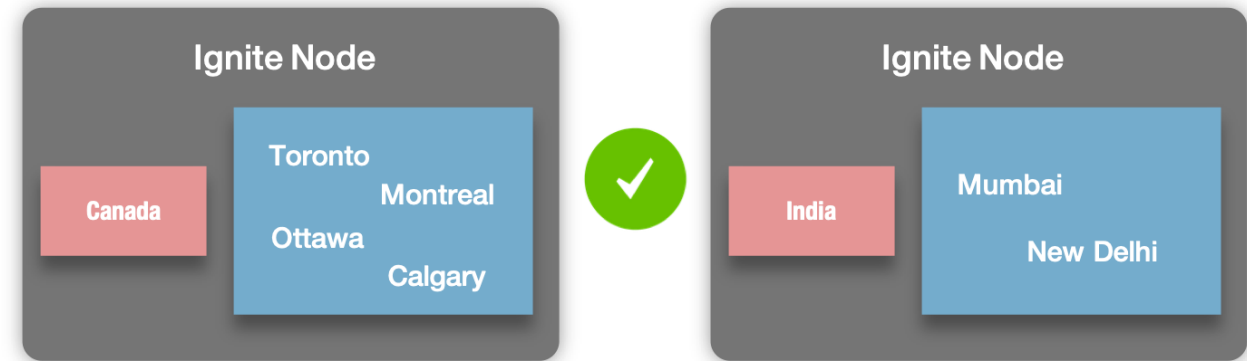


Distributed SQL

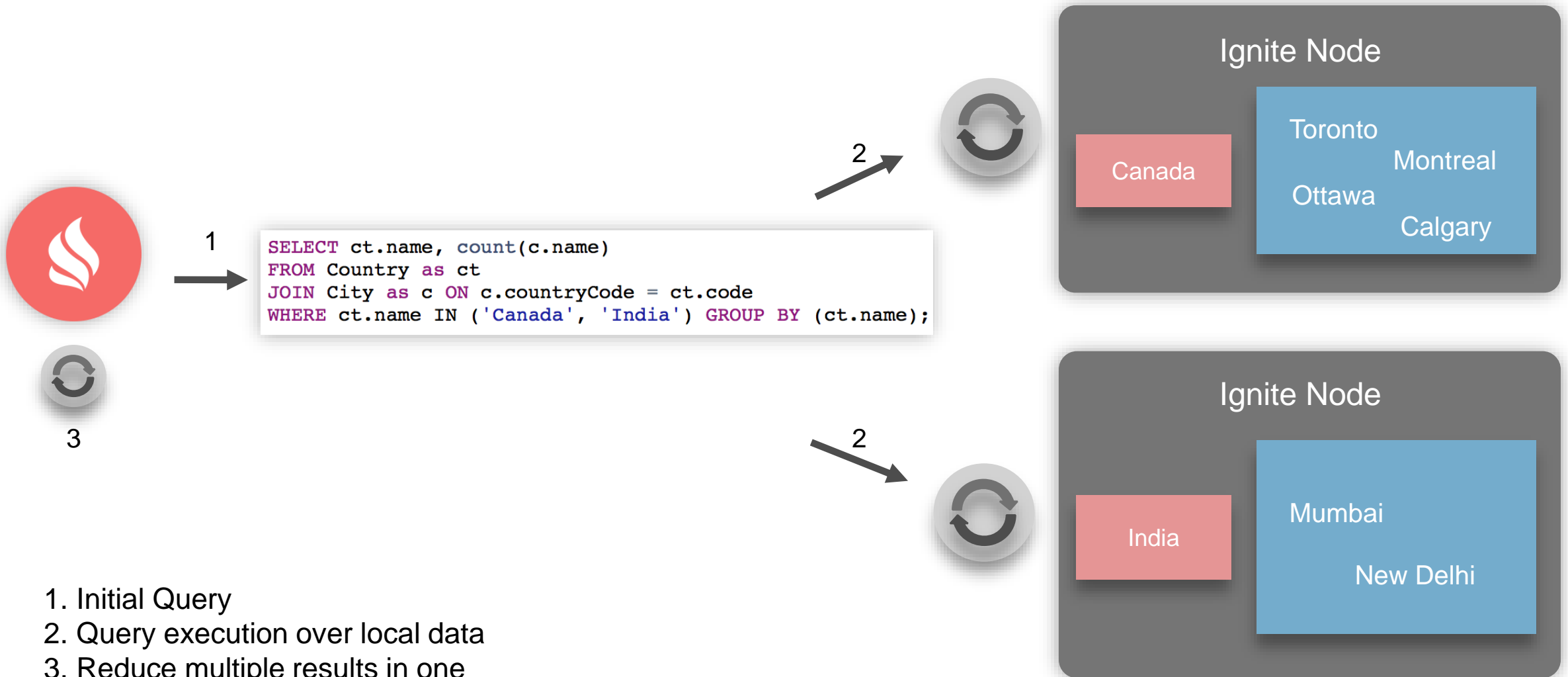


Holy Grail of Distributed World: Affinity Collocation

- Related data is on the same node
 - Countries and Cities
 - Departments and Employees
- Collocated Processing
 - Efficient Distributed JOINS
 - Collocated Computations
 - Reduced network traffic
 - Performance boost!

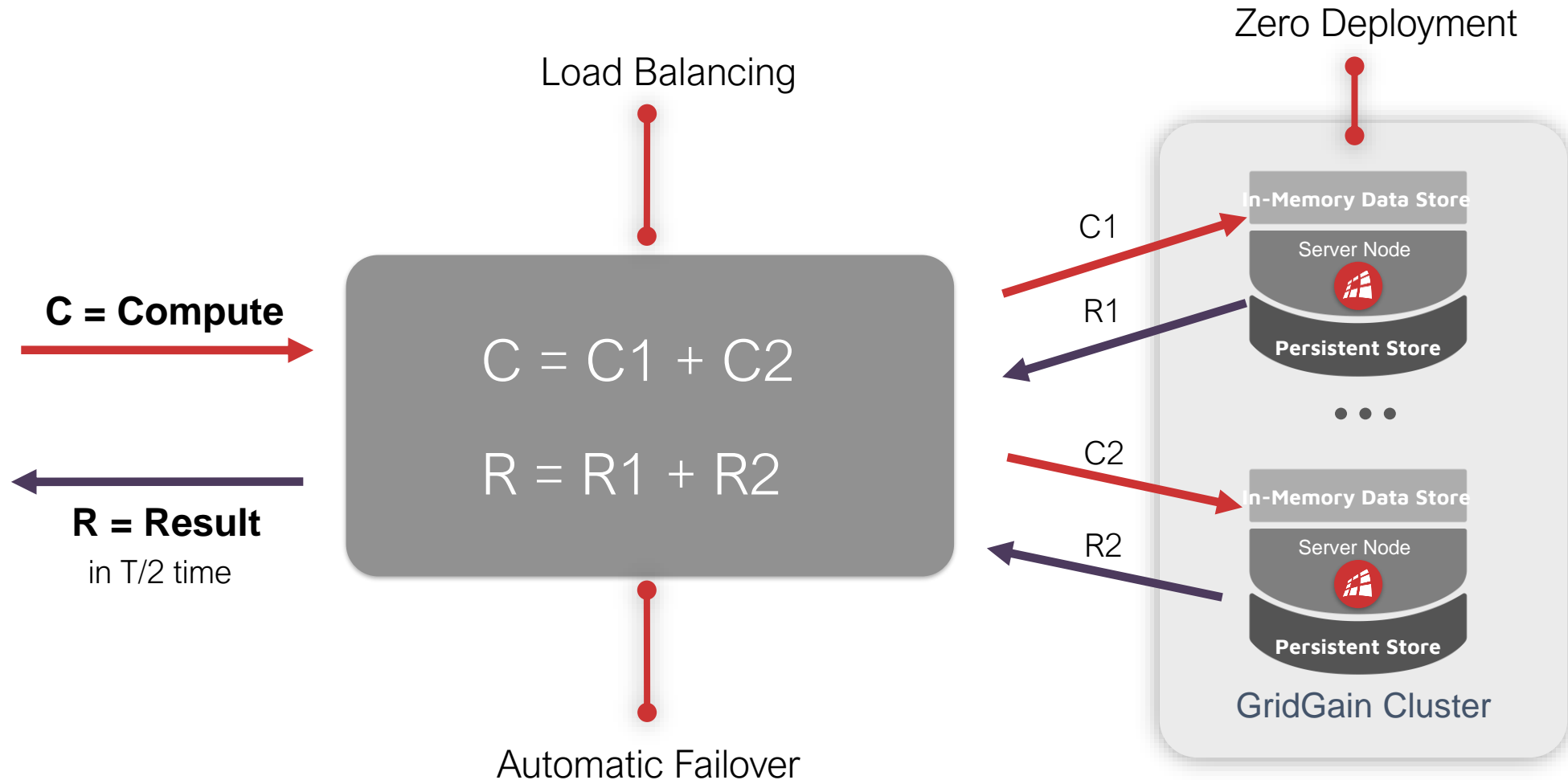


Ignite SQL Queries



1. Initial Query
2. Query execution over local data
3. Reduce multiple results in one

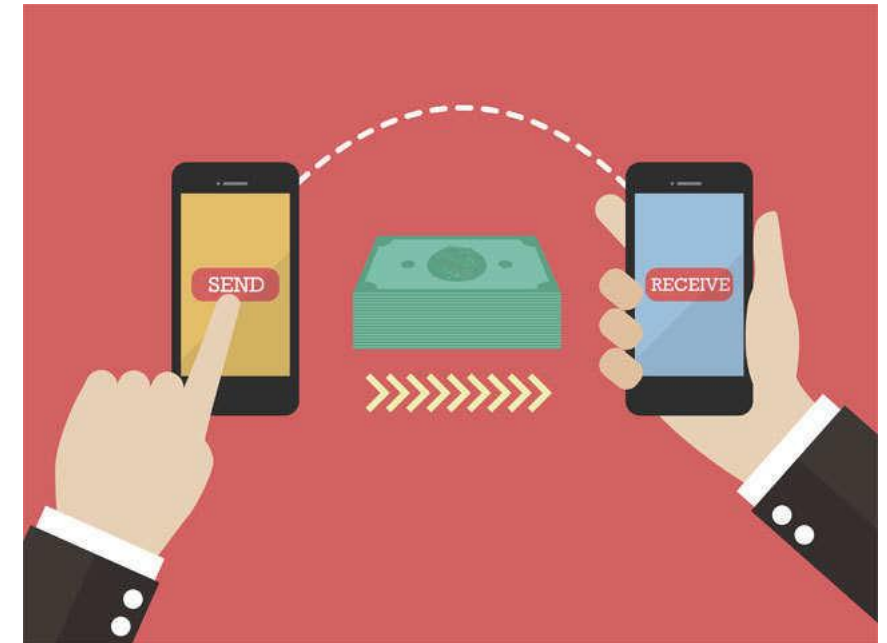
Life Without Stored Procedures: Compute Grid



Transactions and Consistency



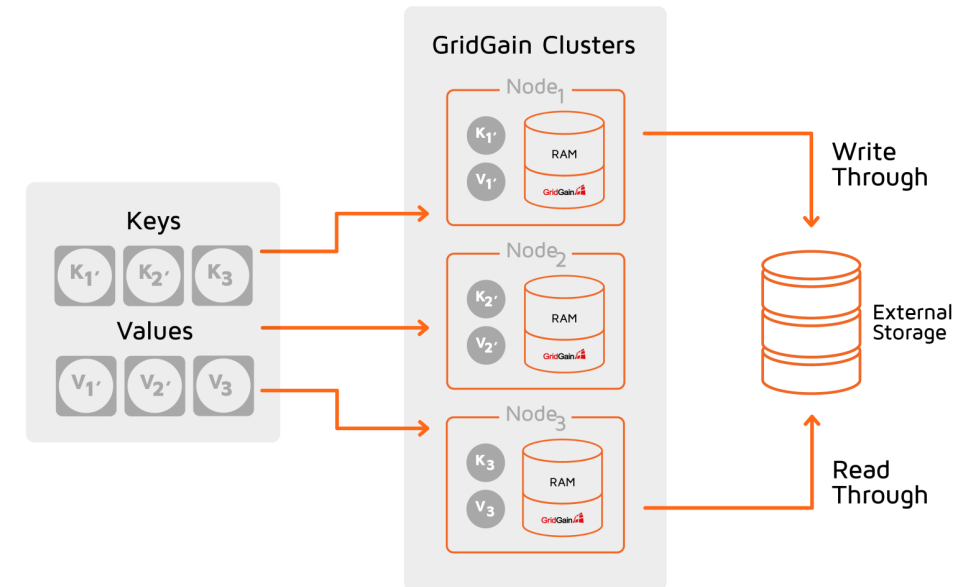
- Distributed Key-Value Transactions
 - 2 phase commit protocol
 - Spans to Postgres
- Transactional SQL (Beta)
 - MVCC
- Strong or relaxed consistency
 - Atomic and transactional tables
 - Tunable Write-ahead-log settings



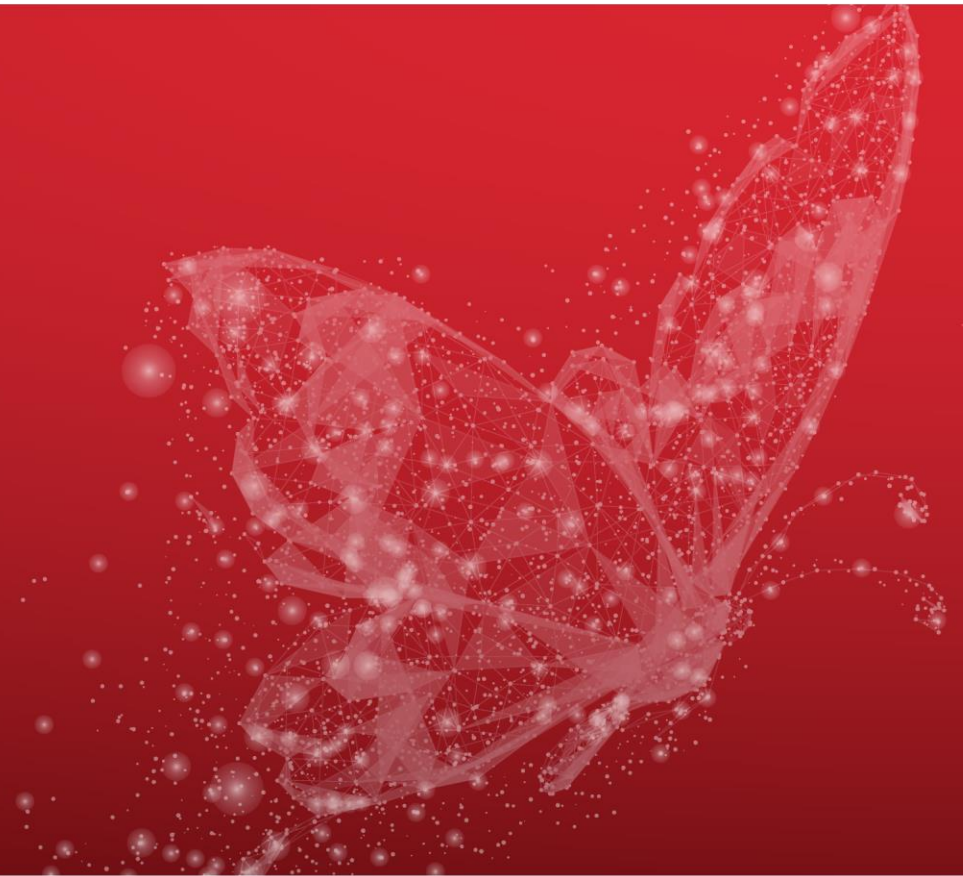
Consistency Across Postgres and Ignite/GridGain



- Coordinator writes to the database first
- Commits in the cluster afterwards
- The database must be transactional
 - Postgres!



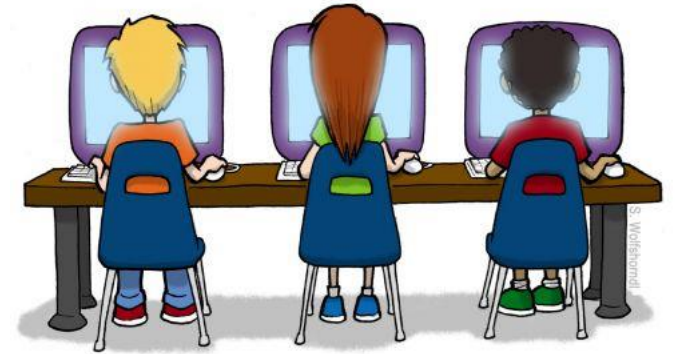
Demo



Apache Ignite – We're Hiring ;)



- Rapidly Growing Community
- Great Way to Learn Distributed Storages, Computing, SQL, ML, Transactions
- How To Contribute:
 - <https://ignite.apache.org/>



Apache Ignite Is a Top 5 Apache Project



A Top 5 Apache Software Foundation Project

Top 5 Dev Mailing Lists

1.  beam

2.  Ignite

3.  kafka

4.  Apache Tomcat

5.  James
Enterprise Mail Server

Top 5 User Mailing Lists

1.  Flink

2.  Lucene

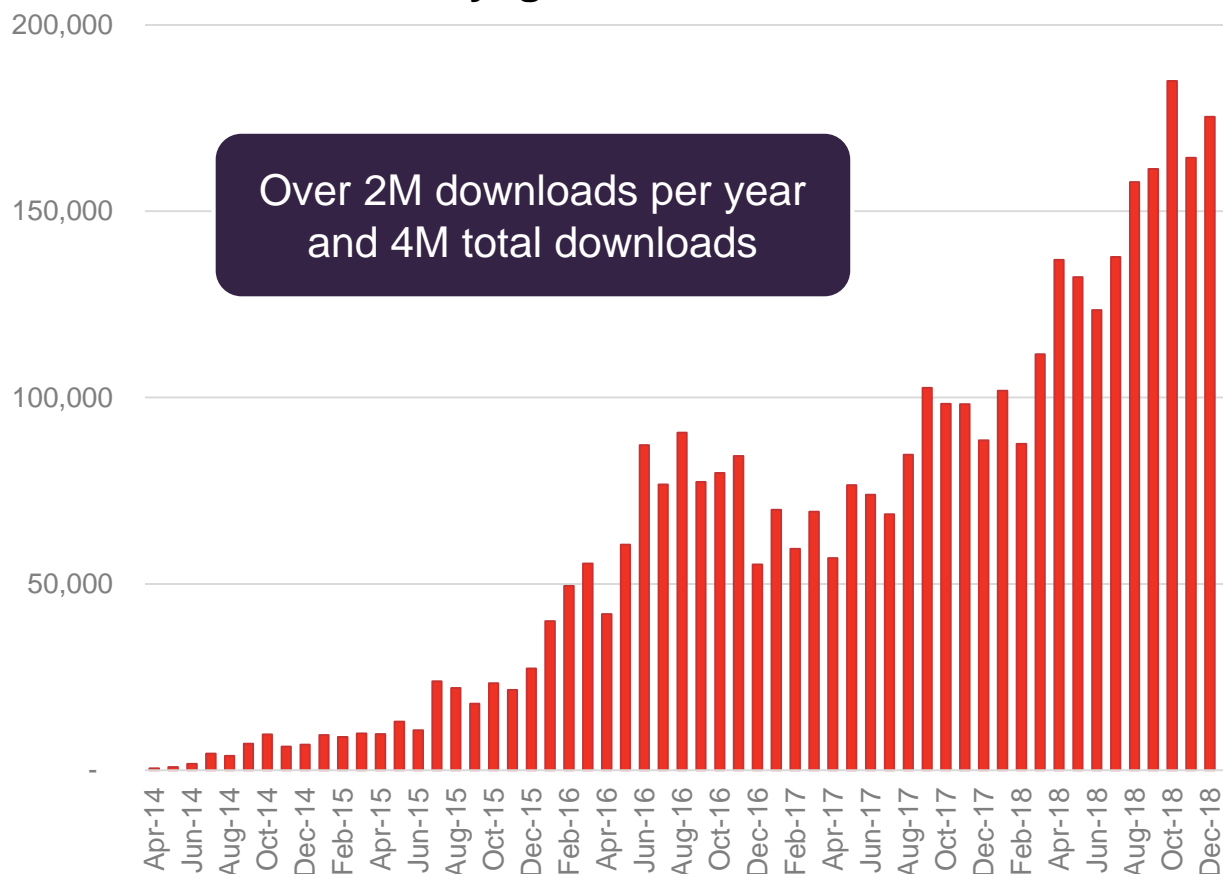
3.  Ignite

4.  cassandra

5.  kafka

From January 1, 2019 Apache Software Foundation Blog Post:
"Apache in 2018 – By The Digits"

Monthly Ignite/GridGain Downloads



Q&A

@apacheignite
@gridgain

