FI

F1

F1, built on Spanner, is "…a rewrite of Google's advertising backend. F1 uses five replicas spread across the United States. Most other applications will probably replicate their data across 3 to 5 datacenters in one geographic region, but with relatively independent failure modes. That is, most applications will choose lower latency over higher availability, as long as they can survive 1 or 2 datacenter failures." (Corbett et al. 2013, 8:2).

"[Adwords] was originally based on a MySQL database that was manually sharded many ways. The uncompressed dataset is tens of terabytes, which is small compared to many NoSQL instances, but was large enough to cause difficulties with sharded MySQL." (Corbett et al. 2013, 8:17).

F1

"The MySQL sharding scheme assigned each customer and all related data to a fixed shard. This layout enabled the use of indexes and complex query processing on a per-customer basis, but required some knowledge of the sharding in application business logic. Resharding this revenue-critical database as it grew in the number of customers and the size of their data was extremely costly. The last resharding took over two years of intense effort, and involved coordination and testing across dozens of teams to minimize risk. This operation was too complex to do regularly: as a result, the team had to limit growth on the MySQL database by storing some data in external Bigtables, which compromised transactional behavior and the ability to query across all data." (Corbett et al. 2013, 8:17).

F1

F1 team chose to use Spanner for several reasons:

- Spanner removes the need to manually reshard
- Spanner provides synchronous replication and automatic failover.
  - With MySQL, failover was difficult and risked data-loss and downtime.
- F1 requires strong transactional semantics (transactions across arbitrary data, and consistent reads): not practical in other NoSQL systems.

F1 team also needed secondary indexing on their data. Spanner doesn't yet provide automatic support for secondary indexes, but F1 team could implement their own consistent global indexes using Spanner transactions.

(Corbett et al. 2013, 8:17).

Proceedings of the VLDB Endowment, 6.11: 1068-1079, 2013.

F1: A Distributed SQL Database That Scales

Jeff Shute
Chad Whipkey
David Menestrina
Radek Vingralek
Eric Rollins
Stephan Ellner
Traian Stancescu
Bart Samwel
Mircea Oancea
John Cieslewicz
Himani Apte
Ben Handy
Kyle Littlefield
Ian Rae*

Google, Inc.
*University of Wisconsin-Madison

ABSTRACT

F1 is a distributed relational database system built at Google to support the AdWords business. F1 is a hybrid database that combines high availability, the scalability of NoSQL systems like Bigtable, and the consistency and usability of traditional SQL databases. F1 is built on Spanner, which provides synchronous cross-datacenter replication and strong consistency. Synchronous replication implies higher commit latency, but we mitigate that latency by using a hierarchical schema model with structured data types and through smart application design. F1 also includes a fully functional distributed SQL query engine and automatic change tracking and publishing.

1. INTRODUCTION

F1[1] is a fault-tolerant globally-distributed OLTP and OLAP database built at Google as the new storage system for Google's AdWords system. It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements.

The key goals of F1's design are:

1. **Scalability**: The system must be able to scale up, trivially and transparently, just by adding resources.
2. **Consistency and correctness**: The system must provide consistent and correct data.
3. **Usability**: The system must provide full SQL query support and other functionality users expect from a SQL database. Features like indexes and ad hoc query are not just nice to have, but absolute requirements for our business.

Designing applications to cope with concurrency anomalies in their data is very error-prone, time-consuming, and ultimately not worth the performance gains.

Recent publications have suggested that these design goals are mutually exclusive [5, 11, 23]. A key contribution of this paper is to show how we achieved all of these goals in F1's design, and where we made trade-offs and sacrifices. The name F1 comes from genetics, where a *Filial 1 hybrid* is the first generation offspring resulting from a cross mating of distinctly different parental types. The F1 database system is indeed such a hybrid, combining the best aspects of traditional relational databases and scalable NoSQL systems like Bigtable [6].

F1 is built on top of Spanner [7], which provides extremely scalable data storage, synchronous replication, and strong consistency and ordering properties. F1 inherits those fea-

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

F1

OnLine Transaction Processing

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

F1

OnLine Analytical Processing

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

- Scalability: scale up, trivially and transparently, just by adding resources.

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

- Scalability: scale up, trivially and transparently, just by adding resources.
- Availability: system must never go down.

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

- Scalability: scale up, trivially and transparently, just by adding resources.
- Availability: system must never go down.
- Consistency: provide ACID transactions.

F1

"F1 is a fault-tolerant globally d and OLAP database built at Google… It was designed to r MySQL implementation that was not able to meet our grov reliability requirements". (Shute et al. 2013, p.1).

Atomic
Consistent
Isolated
Durable
(Lecture 10)

Design Goals:

- Scalability: scale up, trivially a transparently, just by adding resources.
- Availability: system must ne er go down.
- Consistency: provide ACID transactions.

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

- Scalability: scale up, trivially and transparently, just by adding resources.
- Availability: system must never go down.
- Consistency: provide ACID transactions.
- Usability: full SQL support.

F1

"F1 is a fault-tolerant globally distributed OLTP and OLAP database built at Google… It was designed to replace a sharded MySQL implementation that was not able to meet our growing scalability and reliability requirements". (Shute et al. 2013, p.1).

Design Goals:

- Scalability: scale up, trivially and transparently, just by adding resources.
- Availability: system must never go down.
- Consistency: provide ACID transactions.
- Usability: full SQL support.

F1 running Adwords since early 2012: 100s of applications; 1000s of users; DB >100Tb; serves >100,000s requests/s; scans >10,000,000,000,000 rows.

F1 Basic Architecture

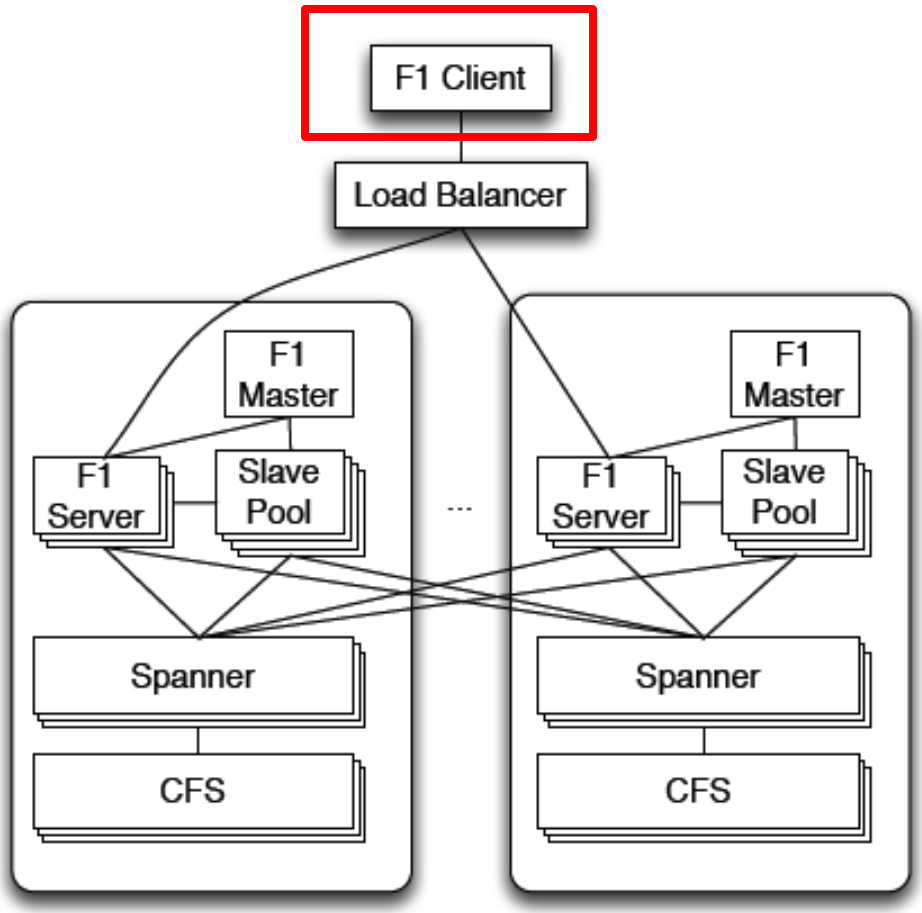

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

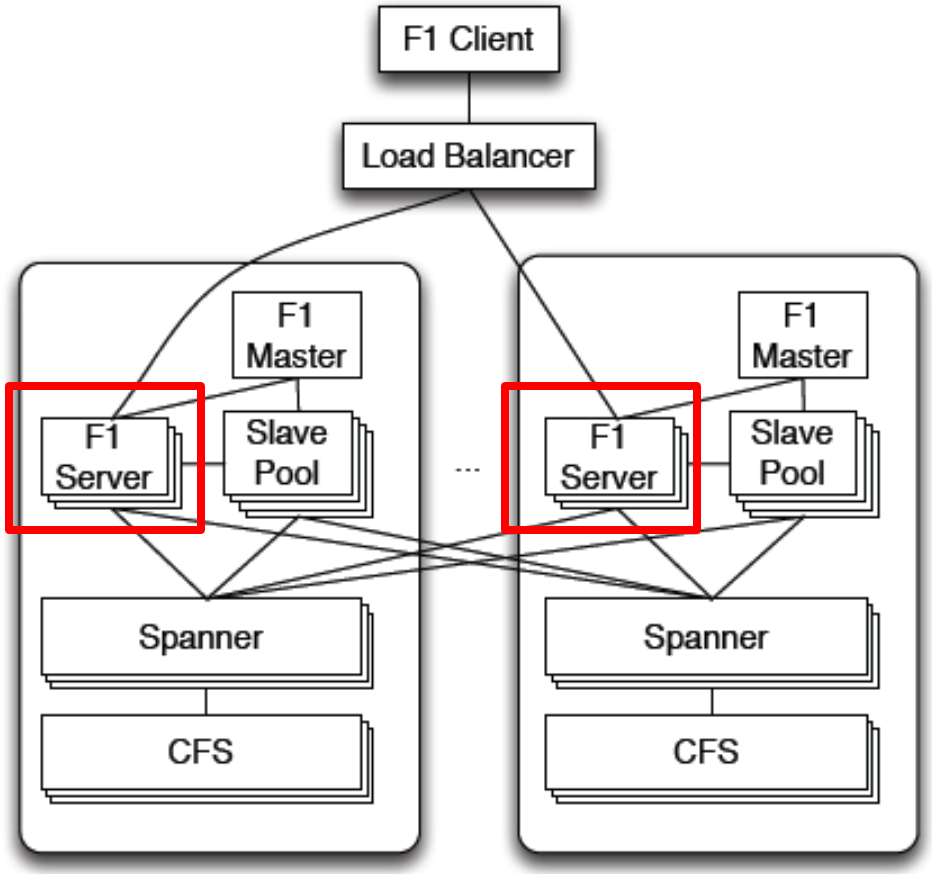

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

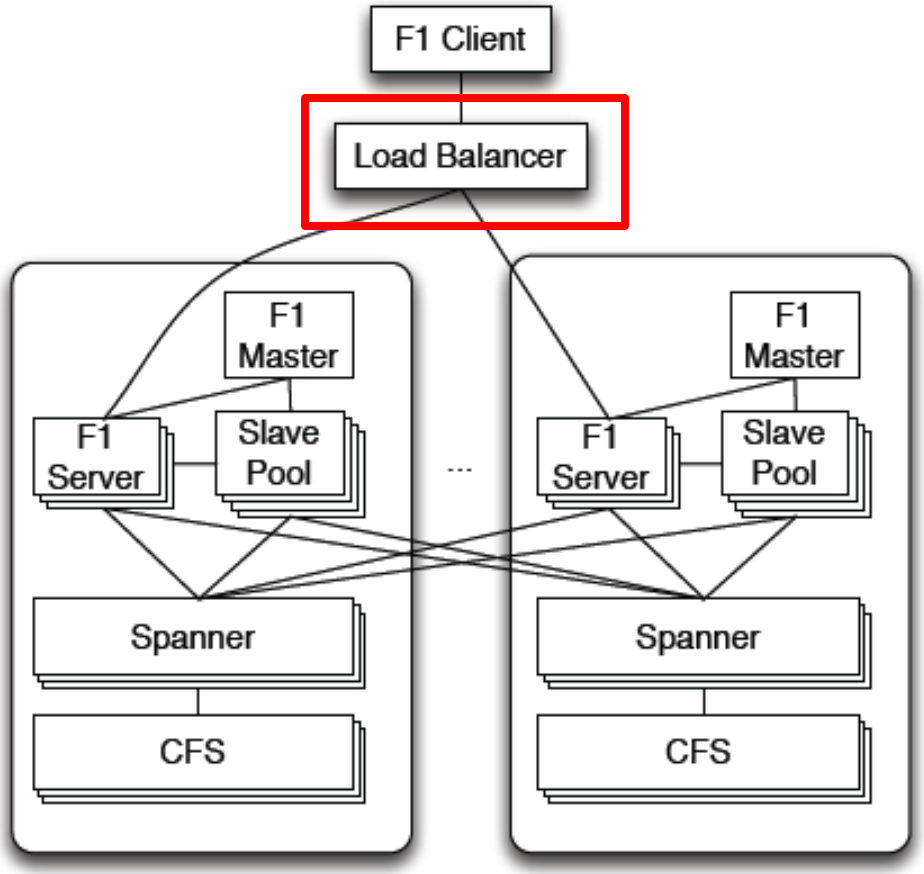

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

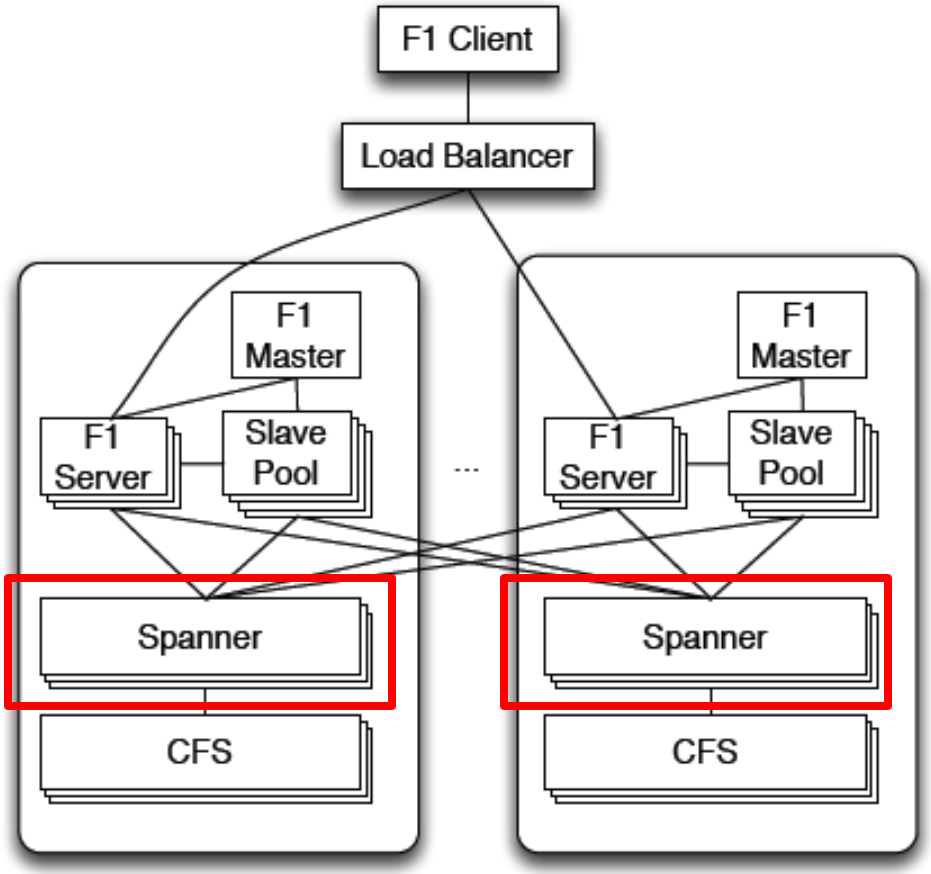

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

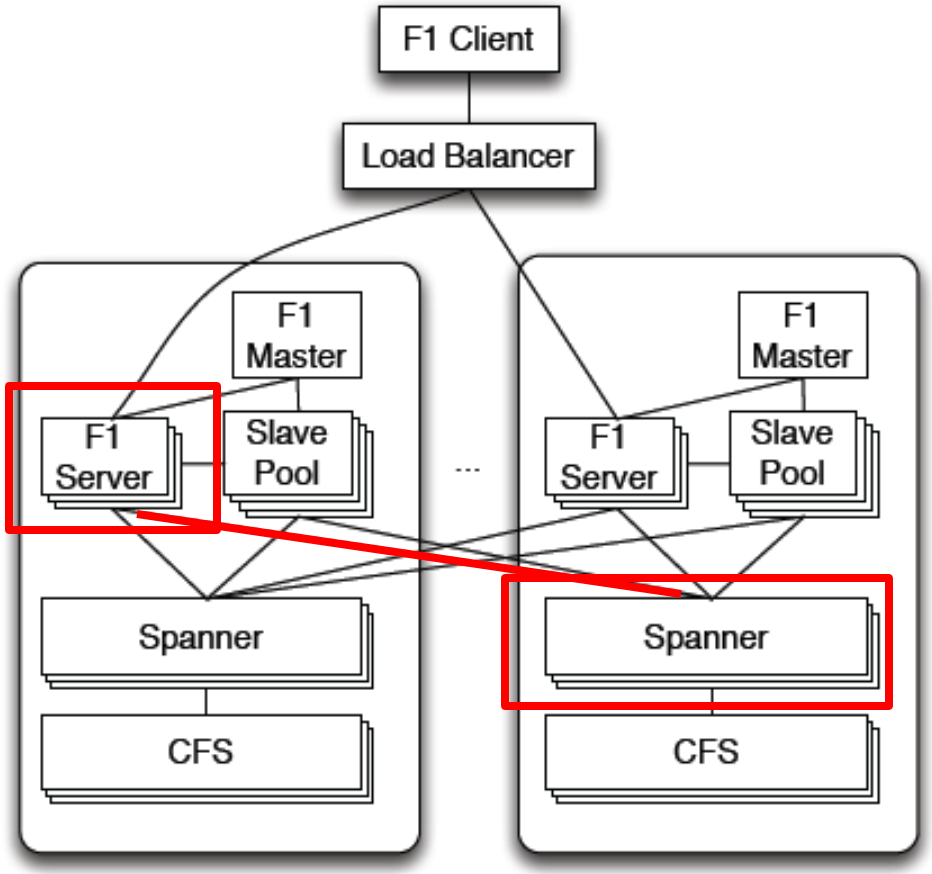

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

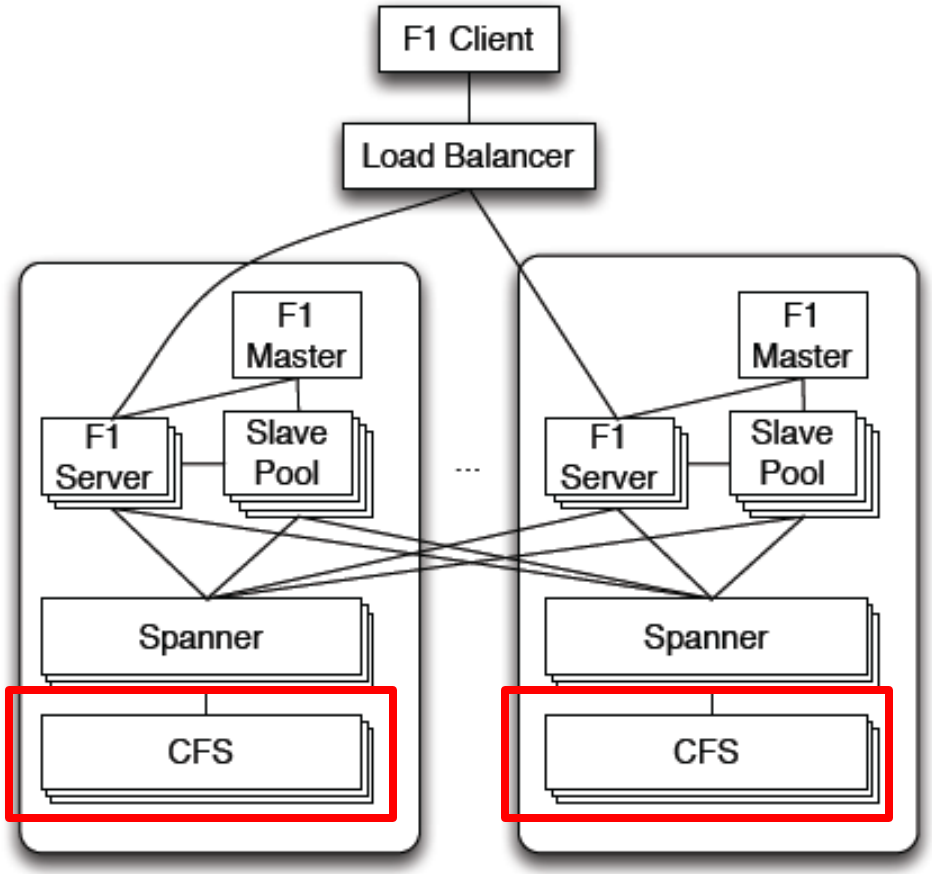

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

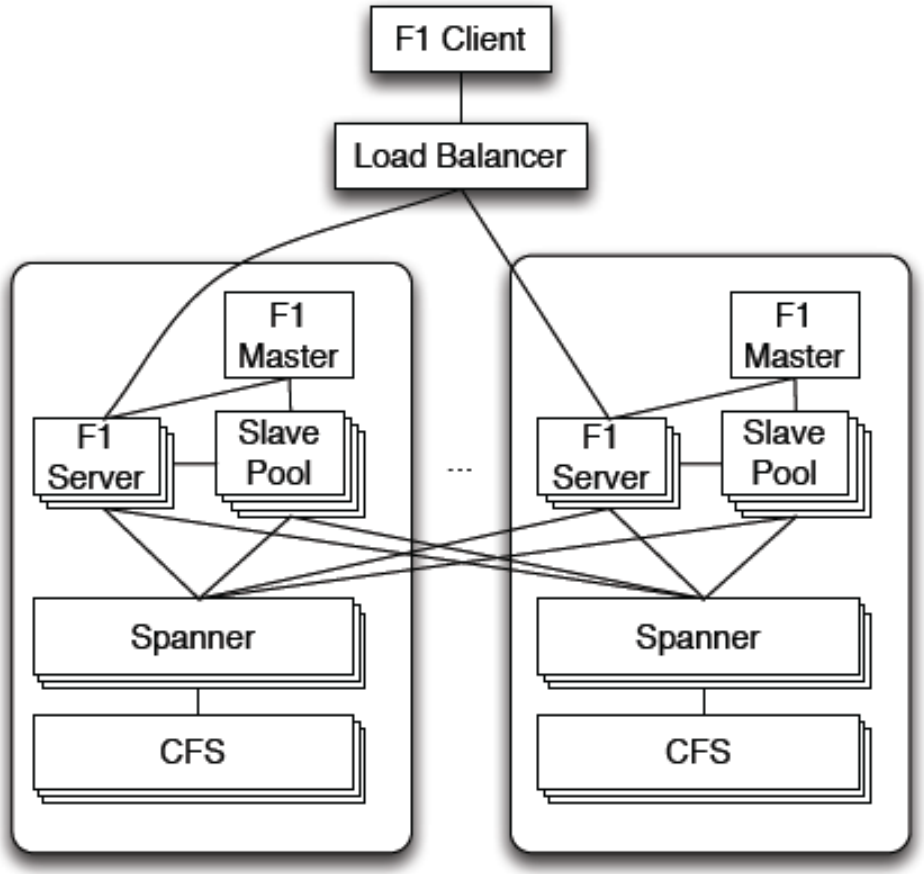

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

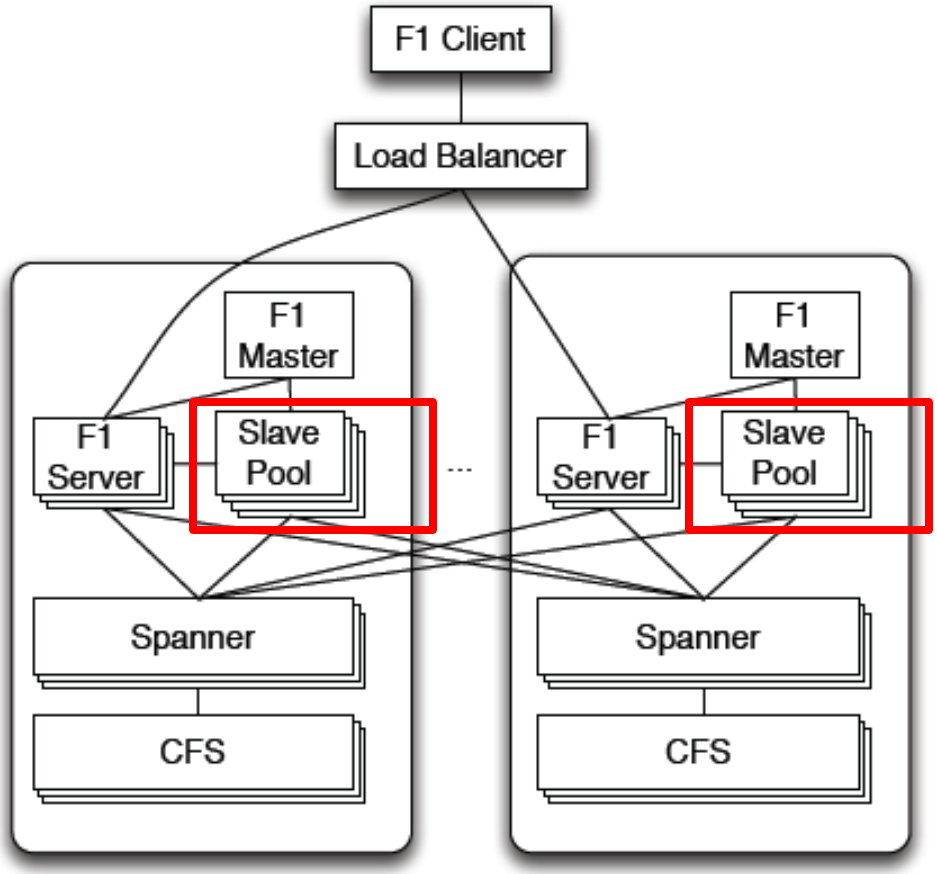

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

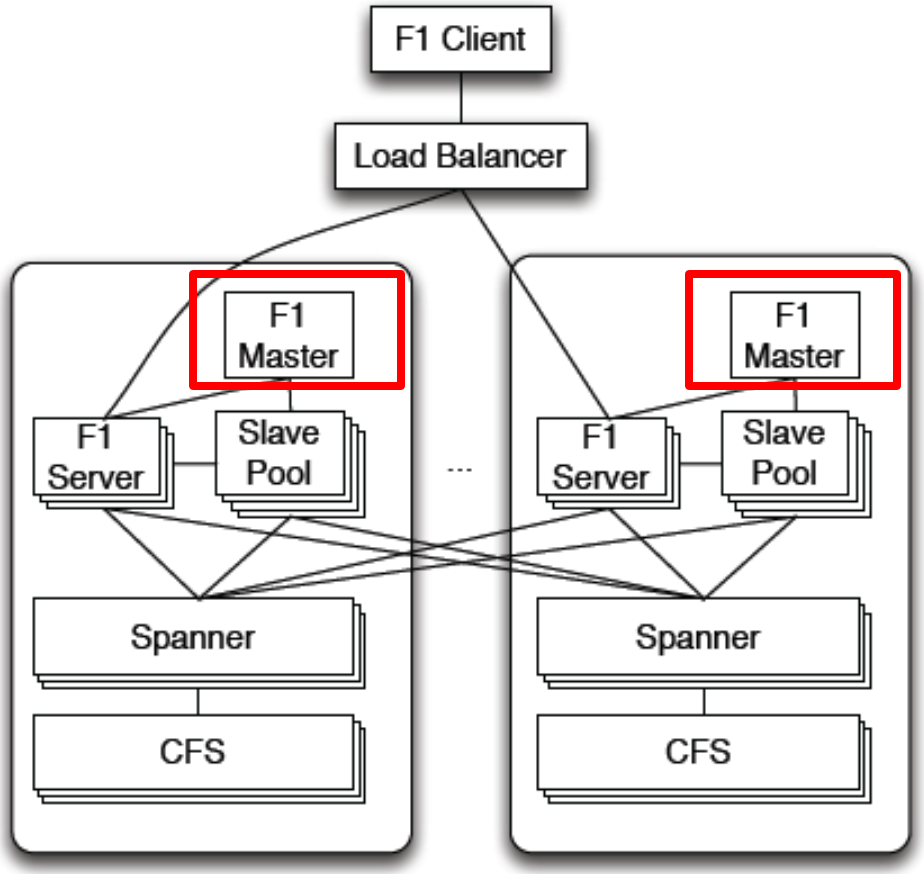

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Basic Architecture

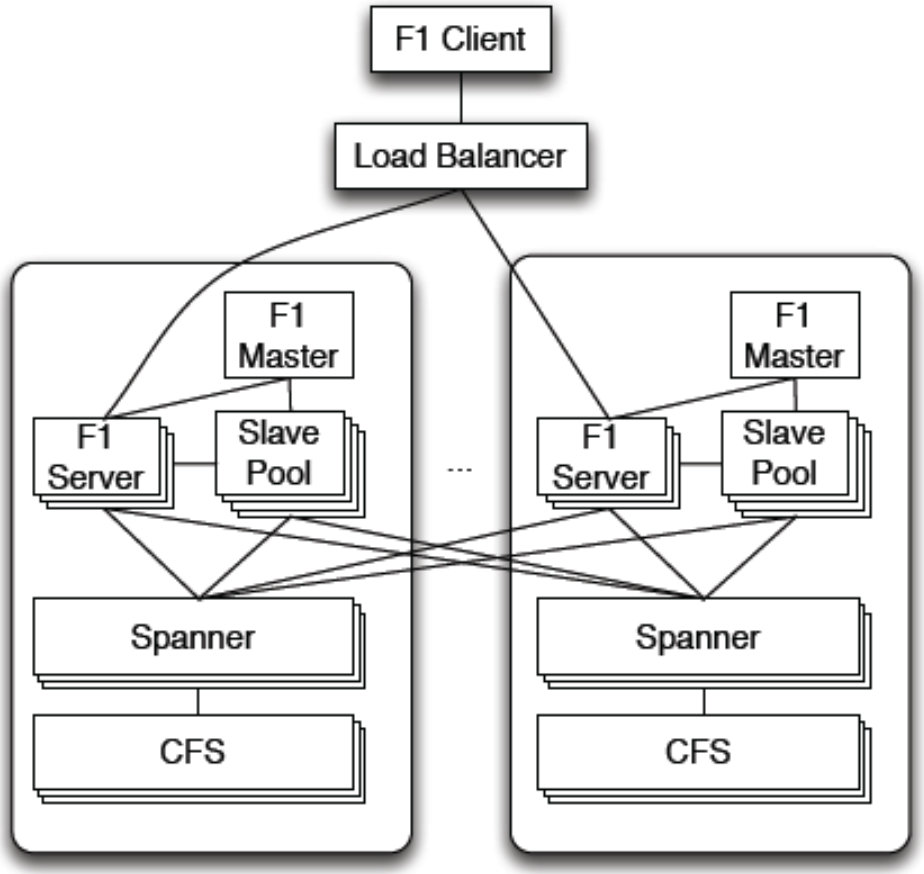

Figure 1: The basic architecture of the F1 system, with servers in two datacenters.

F1 Data Model

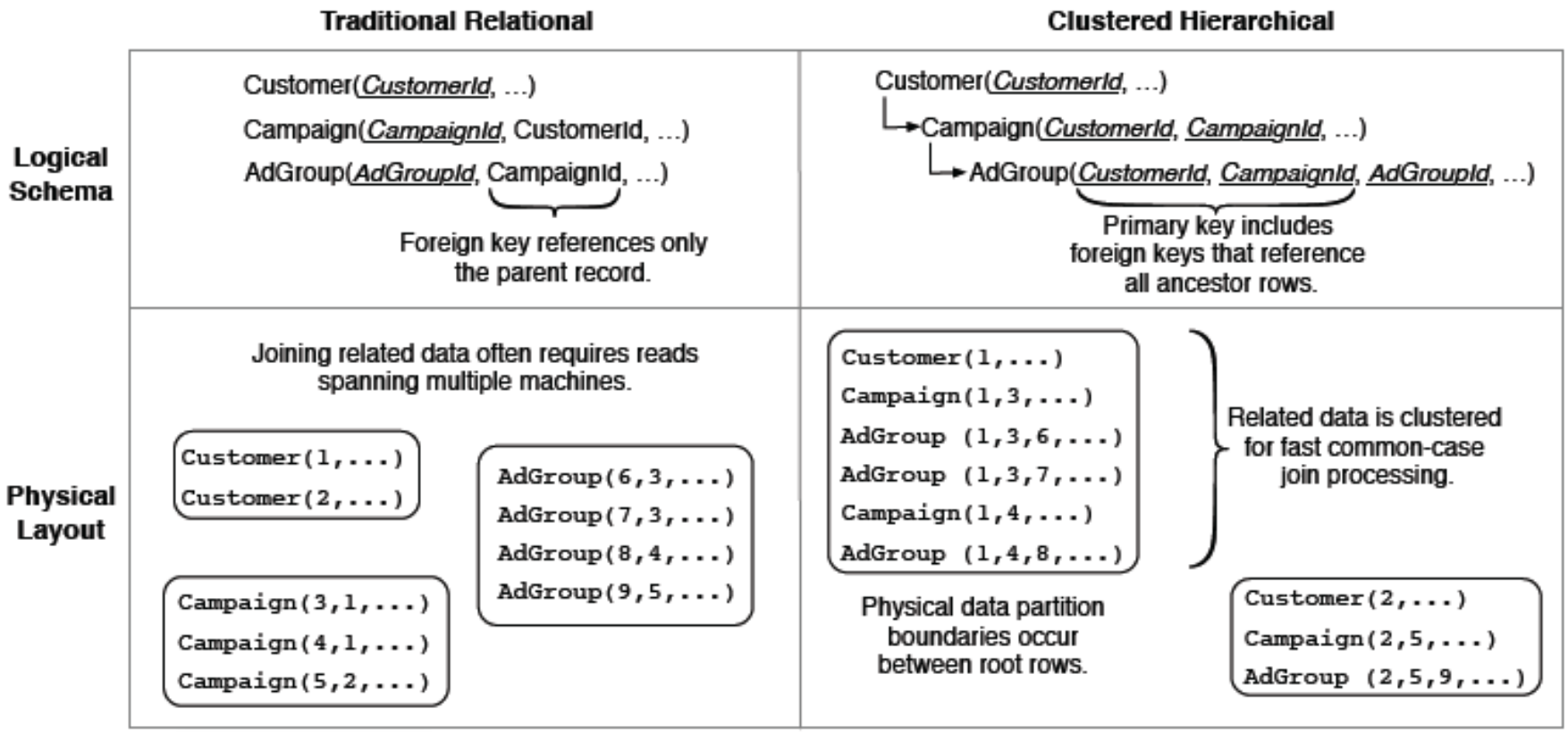

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

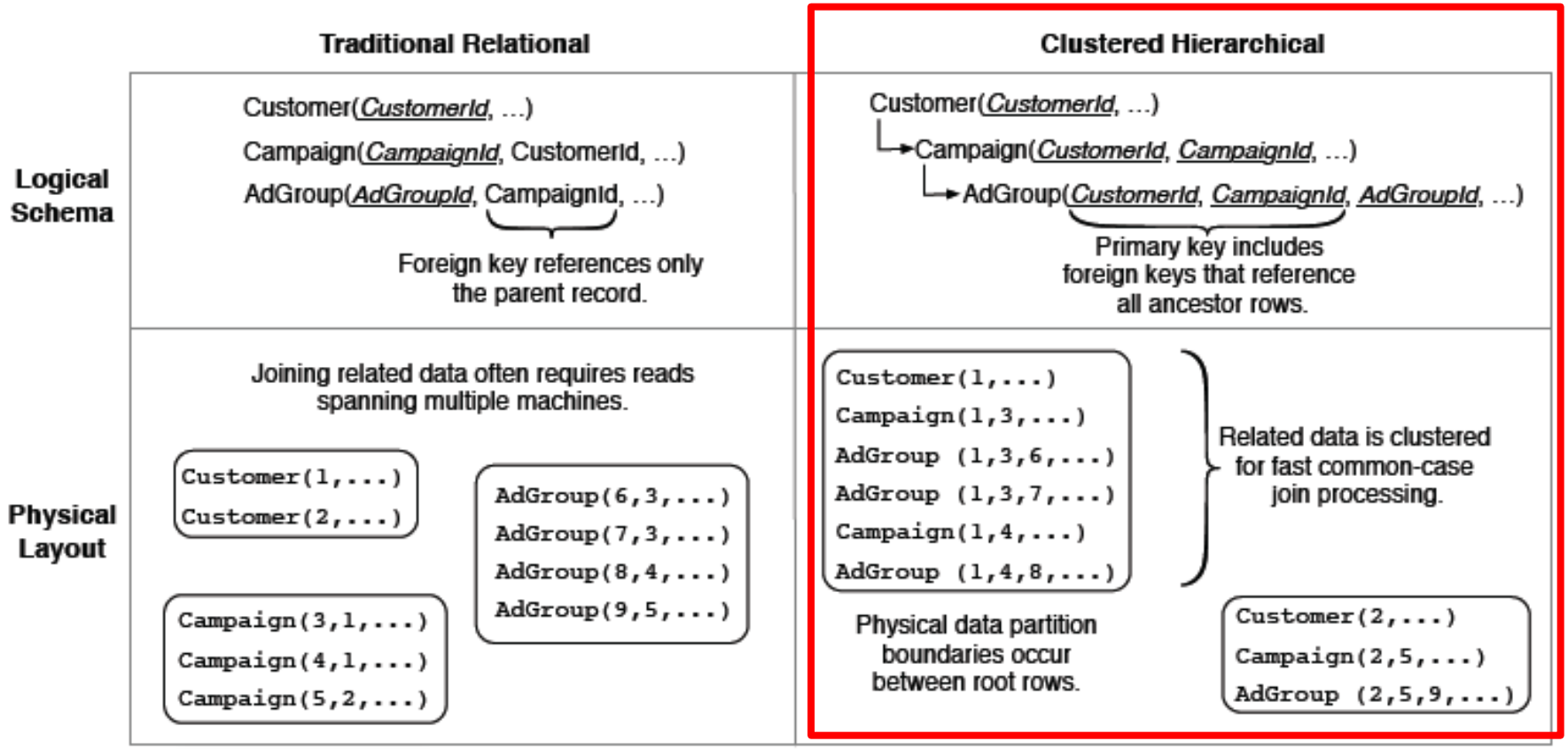

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

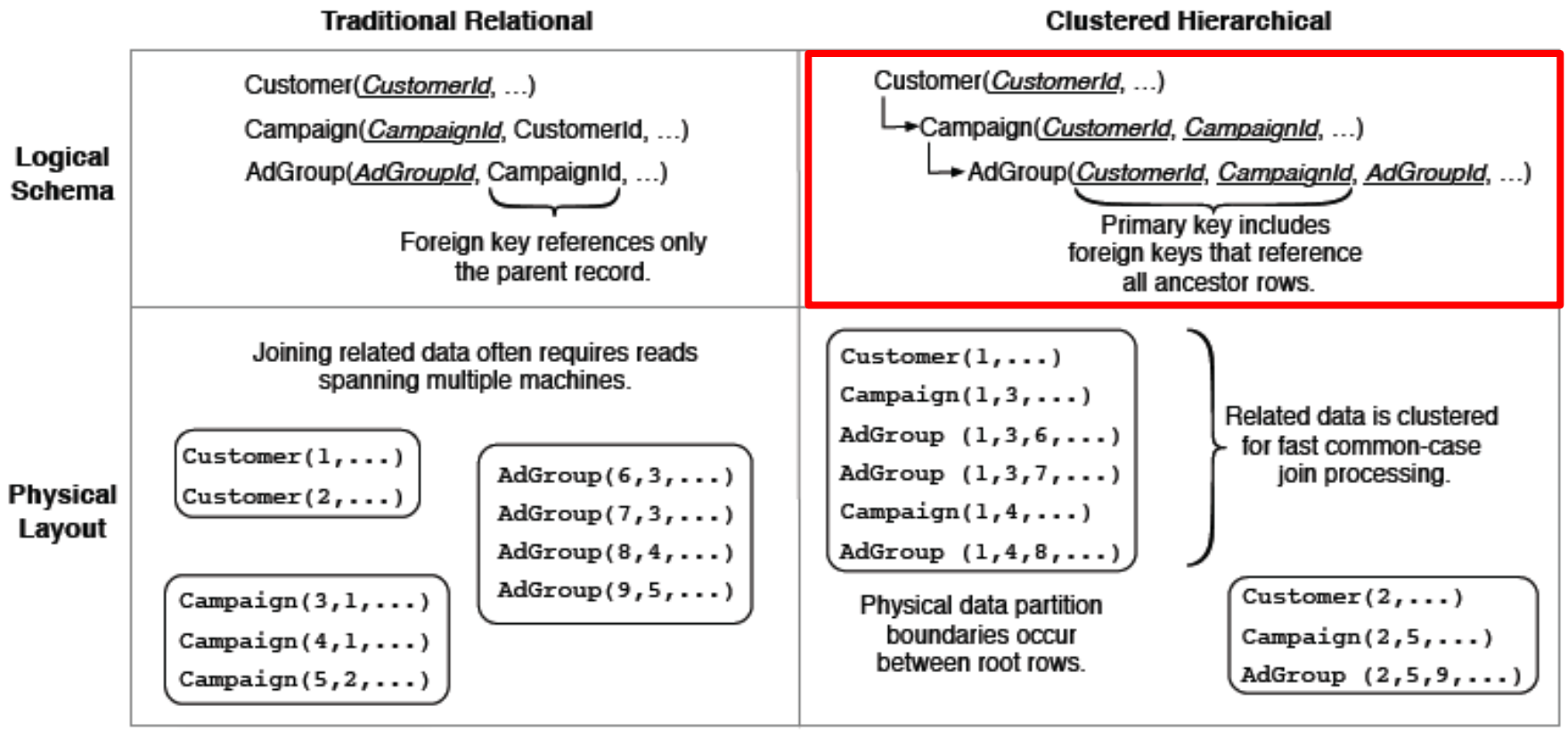

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

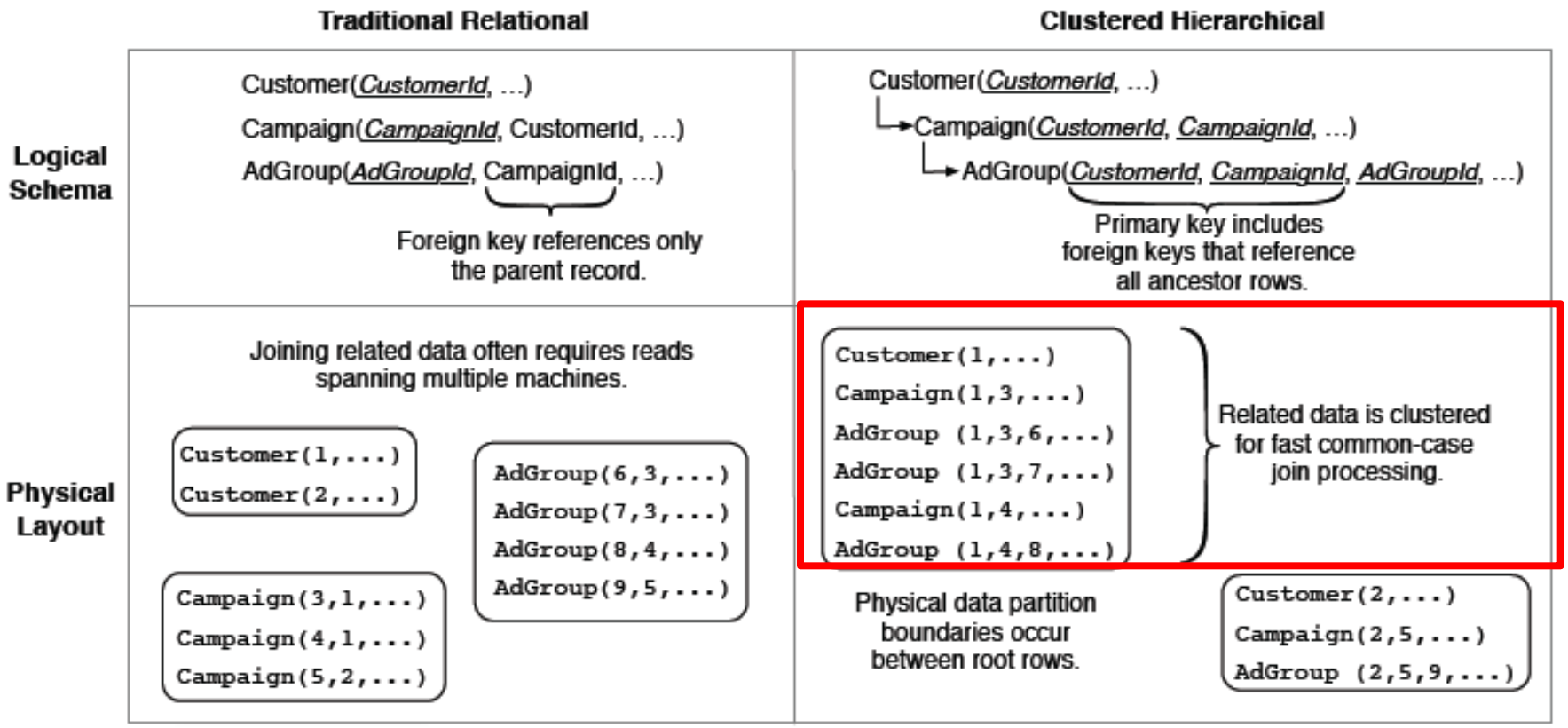

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

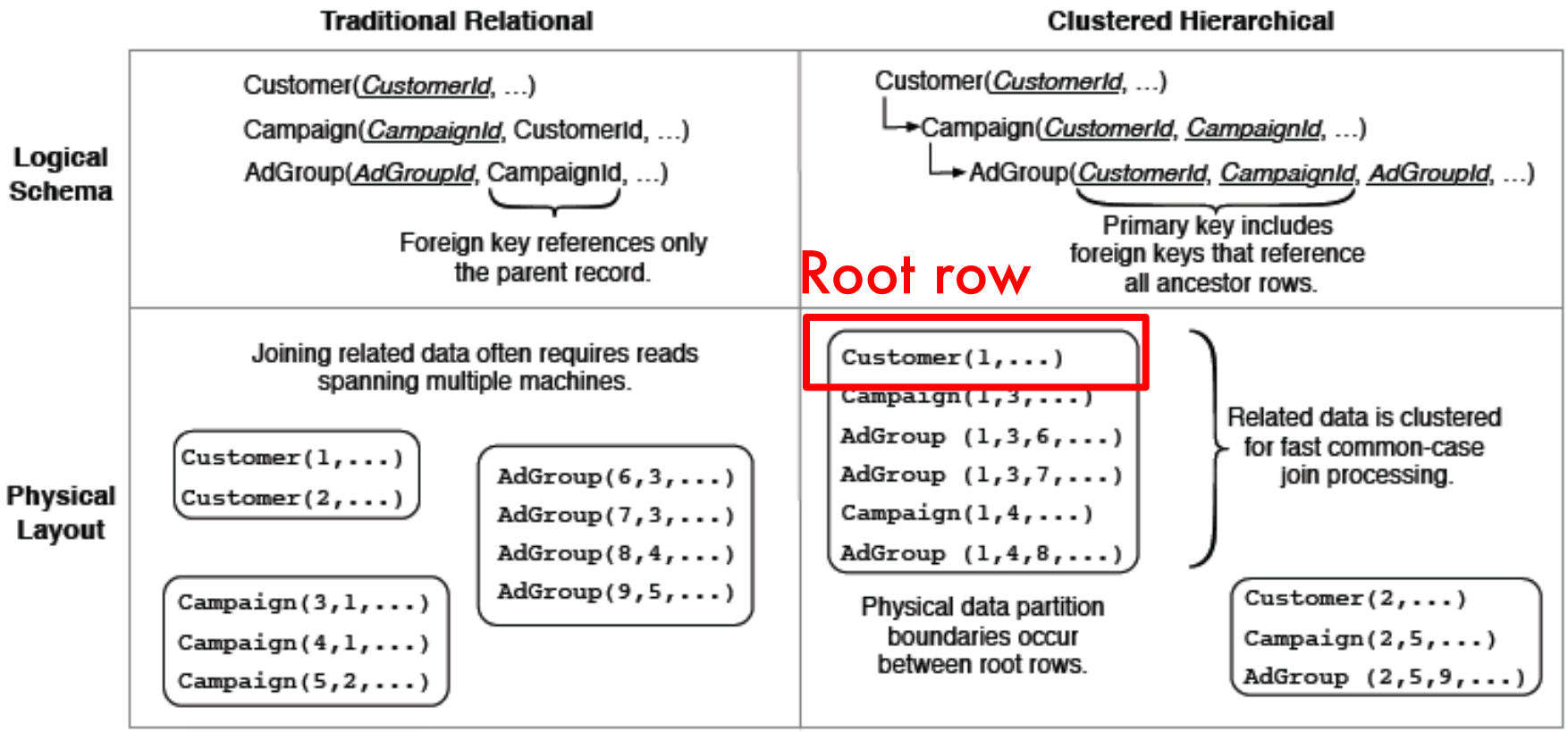

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

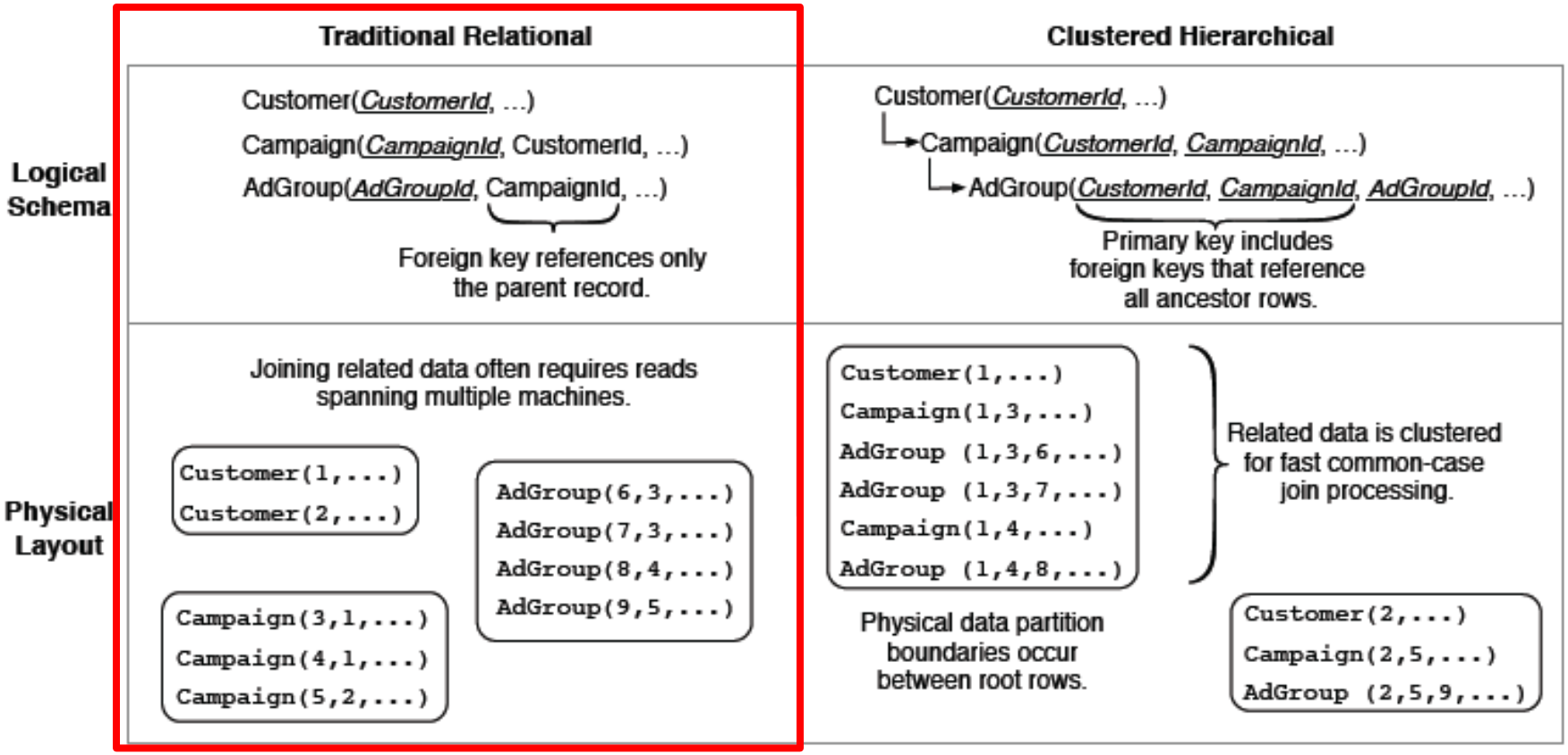

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

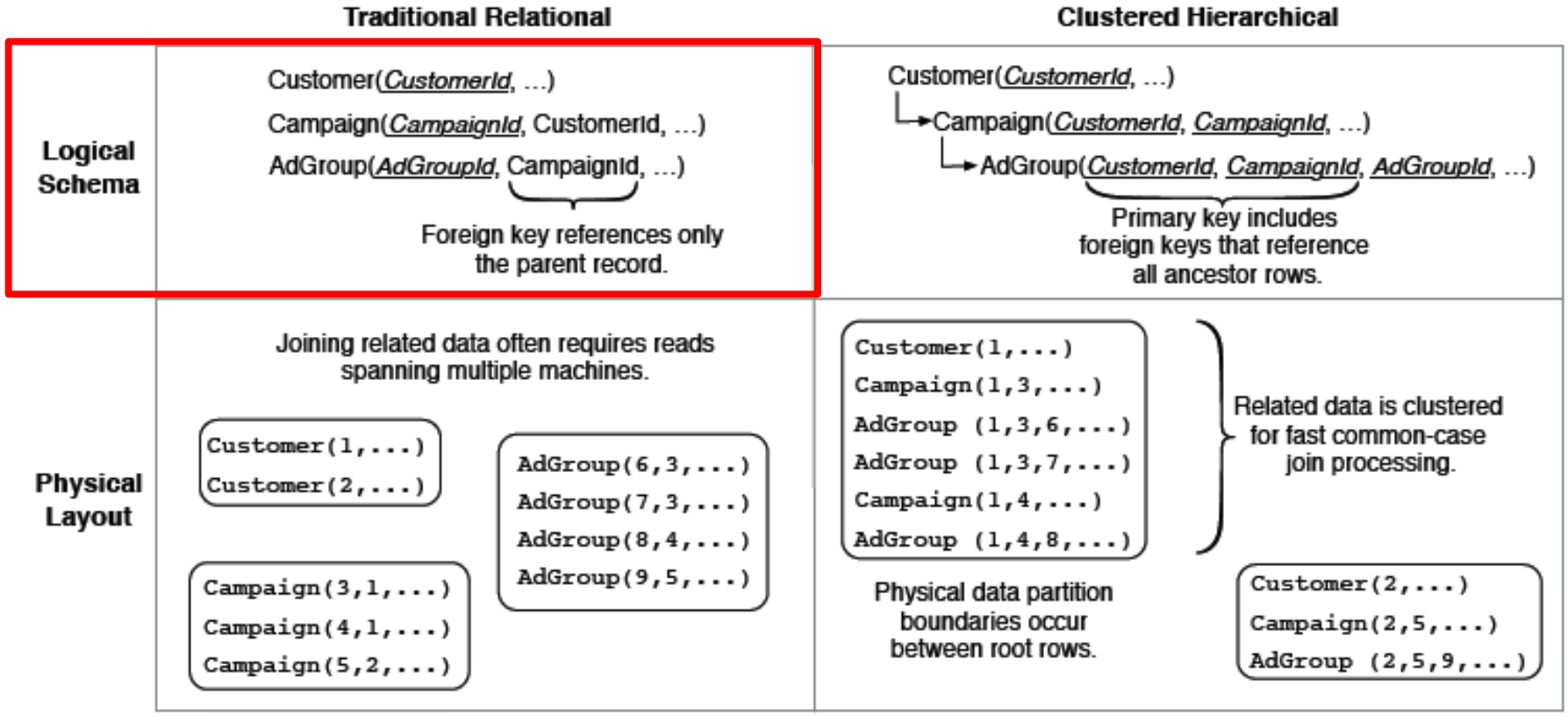

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

F1 Data Model

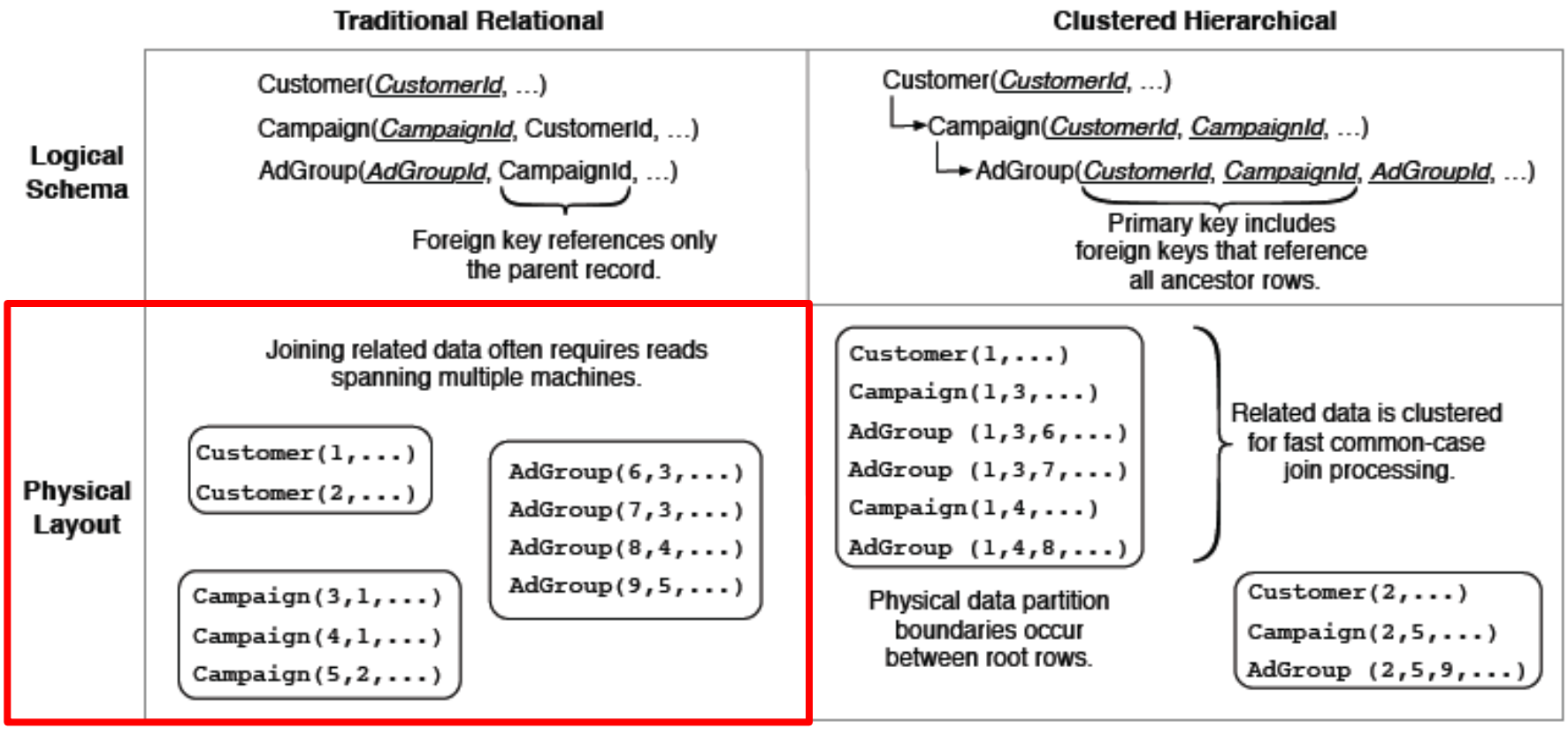

Figure 2: The logical and physical properties of data storage in a traditional normalized relational schema compared with a clustered hierarchical schema used in an F1 database.

Google Core Technology: Summary

In this lecture we've done a very rapid tour through…

- MapReduce
- GFS
- BigTable
- Spanner
- F1

In Lecture 15 we look at the open source equivalents of MapReduce, GFS, and BigTable, all developed by Apache Software Foundation.

It seems highly unlikely that Apache will ever set up projects directly aimed at developing open-source versions of Spanner, at least until there is widespread adoption of TrueTime and the associated GLPS/atomic-clock technology.

And, because F1 is built on Spanner (+ TrueTime), direct open-source version of F1 is unlikely, but other "NewSQL" database technologies are under development.