Diamond: Automating Data Management and Storage for Wide-area, Reactive Applications

Irene Zhang    Niel Lebeck    Pedro Fonseca
Brandon Holt    Raymond Cheng
Ariadna Norberg    Arvind Krishnamurthy    Henry M. Levy

UNIVERSITY of WASHINGTON
Diamond: Automating Data Management and Storage for Wide-area, Reactive Applications

Irene Zhang    Niel Lebeck    Pedro Fonseca
Brandon Holt    Raymond Cheng
Ariadna Norberg    Arvind Krishnamurthy    Henry M. Levy
Reactive applications automatically propagate updates across mobile devices and the cloud.
Reactive applications automatically propagate updates across mobile devices and the cloud.
Reactive applications **automatically** propagate updates across mobile devices and the cloud.
Reactive applications automatically propagate updates across mobile devices and the cloud.
Reactive applications automatically propagate updates across mobile devices and the cloud.
Reactive applications automatically propagate updates across mobile devices and the cloud.
Reactive applications _automatically_ propagate updates across mobile devices and the cloud.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.
Which poses a challenge for app programmers.

This is a complex, distributed data management problem!
Which poses a challenge for app programmers.

This is a complex, distributed data management problem!

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Which poses a challenge for app programmers.

Conclusion: Reactive applications require end-to-end data management with strong guarantees.
Diamond

Diamond is the first reactive data management service, which provides the following guarantees:

• Ensures updates to shared data are consistent and durable

• Coordinates and synchronizes updates reliably across mobile clients and cloud storage

• Automatically triggers application code in response to updates to shared data
Talk Outline

**Diamond System & Programming Model**
What does Diamond provide for reactive apps?

**Diamond Guarantees & Implementation**
What does Diamond guarantee for reactive apps?

**Evaluation**
How does Diamond impact app complexity and performance?
Diamond System Model

Client Devices

- App Process
- libDiamond

Diamond Cloud

- Front-end Servers
- Distributed Key-Value Store
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.
Reactive Data Types (RDTs)
Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings
Reactive Data Types (RDTs)
Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings
**Reactive Data Types (RDTs)**

Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in `libDiamond` language bindings

```
DCounter player1;
DCounter player2;
```
Reactive Data Types (RDTs)
Shared, persistent data structures

- Simple data structures including primitives (e.g., string, long), collections (e.g., list) and Conflict-free Data Types (e.g., counter, set)
- Data type semantics avoid false sharing and enable commutative operations
- Defined in libDiamond language bindings
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

---

**App**
- player1
- player2
- turn

**Store**

**App**
- player1
- player2
- turn

**libDiamond**

**Diamond Cloud**
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

• Key abstraction for providing flexible, shared memory

• Gives apps control over what app data is shared and how it is organized

• Enables Diamond to automatically provide availability, fault-tolerance and consistency to RDTs
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

- Key abstraction for providing flexible, shared memory
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to automatically provide availability, fault-tolerance and consistency to RDTs
Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

• Key abstraction for providing flexible, shared memory

• Gives apps control over what app data is shared and how it is organized

• Enables Diamond to automatically provide availability, fault-tolerance and consistency to RDTs

```c
rmap(player1, "irene");
rmap(player2, "niel");
```
**Reactive Data Map (rmap)**

Binding between RDTs in the app and the Diamond store

- Key abstraction for providing flexible, shared memory
- Gives apps control over what app data is shared and how it is organized
- Enables Diamond to automatically provide availability, fault-tolerance and consistency to RDTs

```plaintext
rmap(player1, "irene");
rmap(player2, "niel");
rmap(turn, "turn");
```
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

---

**App**
- player1
- player2
- turn

**Store**
- irene
- niel
- turn

**App**
- player1
- player2
- turn
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store
Read-write Transactions
Read-write transactions to update shared RDTs.

• Execute application code to update rmapped RDTs
• Gives application programmers control over when to synchronize shared data
• Ensures safe concurrent access to shared data
Read-write Transactions
Read-write transactions to update shared RDTs.

• Execute application code to update rmapped RDTs
• Gives application programmers control over when to synchronize shared data
• Ensures safe concurrent access to shared data
Read-write Transactions
Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data

```
begin();
player1 = 0;
```
**Read-write Transactions**

Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data
**Read-write Transactions**
Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data
Read-write Transactions
Read-write transactions to update shared RDTs.

- Execute application code to update rmapped RDTs
- Gives application programmers control over when to synchronize shared data
- Ensures safe concurrent access to shared data

execute application code to update rmapped RDTs
Gives application programmers control over when to synchronize shared data
Ensures safe concurrent access to shared data

begin();
player1 = 0;
player2 = 0;
turn = “irene”;
commit();
**Diamond Programming Model**

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

---

**Reactive Data Types (RDTs)**

<table>
<thead>
<tr>
<th>App</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>player1</td>
<td>irene 0</td>
</tr>
<tr>
<td>player2</td>
<td>nil 0</td>
</tr>
<tr>
<td>turn</td>
<td>irene</td>
</tr>
</tbody>
</table>

**Diamond Cloud**

libDiamond

---

**Store**

<table>
<thead>
<tr>
<th>player1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>player2</td>
<td>0</td>
</tr>
<tr>
<td>turn</td>
<td>irene</td>
</tr>
</tbody>
</table>

libDiamond
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

---

<table>
<thead>
<tr>
<th>App</th>
<th>Store</th>
<th>App</th>
</tr>
</thead>
<tbody>
<tr>
<td>player1</td>
<td>0</td>
<td>player1</td>
</tr>
<tr>
<td>player2</td>
<td>0</td>
<td>player2</td>
</tr>
<tr>
<td>turn</td>
<td>irene</td>
<td>turn</td>
</tr>
</tbody>
</table>

**libDiamond**

**Diamond Cloud**

**libDiamond**
Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

- Key abstraction for automatically propagating updates to local data
- Gives apps a consistent view of shared data and control over what to sync
- Automatically triggers app code in response to updates from read-write transactions to shared RDTs
Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

• Key abstraction for **automatically propagating updates** to local data

• Gives apps a consistent view of shared data and control over what to sync

• Automatically triggers app code in response to updates from read-write transactions to shared RDTs
Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

- Key abstraction for automatically propagating updates to local data
- Gives apps a consistent view of shared data and control over what to sync
- Automatically triggers app code in response to updates from read-write transactions to shared RDTs
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

---

**App**
- player1: 0
- player2: 0
- turn: irene

**Store**
- irene: 0
- niel: 0
- turn: irene

**Diamond Cloud**

---

**libDiamond**

---

**App**
- player1: 0
- player2: 0
- turn: irene

**libDiamond**

---

**Irene**

---

**Niel**

---

**Pin The Advisor**

---

**Irene**

---

**Niel**
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

You earned 5 pts!
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

You earned 5 pts!

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

Diamond Cloud

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

---

You earned 5 pts!

---

Reactive Transactions

Read-only transactions that re-execute app code when the read set updates.

**Reactive Data Types (RDTs)**

**Reactive Data Map (rmap)**

**Diamond Cloud**

**App**

<table>
<thead>
<tr>
<th>begin</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>player</td>
<td>0</td>
</tr>
<tr>
<td>turn</td>
<td>0</td>
</tr>
<tr>
<td>nil</td>
<td>0</td>
</tr>
</tbody>
</table>

**Store**

<table>
<thead>
<tr>
<th>irene</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>nil</td>
<td>0</td>
</tr>
<tr>
<td>turn</td>
<td>irene</td>
</tr>
</tbody>
</table>

**libDiamond**

**libDiamond**

---

16
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

---

**Diamond Cloud**

<table>
<thead>
<tr>
<th>Store</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>irene</td>
<td>5</td>
</tr>
<tr>
<td>niel</td>
<td>0</td>
</tr>
</tbody>
</table>

**App**

| player1    | 5     |
| player2    | 0     |

**libDiamond**

You earned 5 pts!

---

**Reactive Transactions**

- Reactive Data Types (RDTs)
  - Shared, persistent data structures

- Reactive Data Map (rmap)
  - Binding between RDTs in the app and the Diamond store

- Reactive Transactions
  - Read-only transactions that re-execute app code when the read set updates.

**Reactive Transactions**

- Reactive Data Types (RDTs)
  - Shared, persistent data structures

- Reactive Data Map (rmap)
  - Binding between RDTs in the app and the Diamond store

- Reactive Transactions
  - Read-only transactions that re-execute app code when the read set updates.
Diamond Programming Model

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

Read-write Transactions
Read-write transactions to update shared RDTs.

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

Diamond Cloud

Store

<table>
<thead>
<tr>
<th></th>
<th>Irene</th>
<th>Niel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Turn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

App

<table>
<thead>
<tr>
<th></th>
<th>player1</th>
<th>player2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Turn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

libDiamond

Pin The Advisor

You earned 5 pts!

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

App

<table>
<thead>
<tr>
<th></th>
<th>player1</th>
<th>player2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Turn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Store

<table>
<thead>
<tr>
<th></th>
<th>Irene</th>
<th>Niel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Turn</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

libDiamond
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

---

You earned 5 pts!

---

Reactive Transactions
Read-only transactions that re-execute app code when the read set updates.

Reactive Data Types (RDTs)
Shared, persistent data structures

Reactive Data Map (rmap)
Binding between RDTs in the app and the Diamond store

---

You earned 5 pts!
Diamond Programming Model

**Reactive Data Types (RDTs)**
Shared, persistent data structures

**Reactive Data Map (rmap)**
Binding between RDTs in the app and the Diamond store

**Read-write Transactions**
Read-write transactions to update shared RDTs.

**Reactive Transactions**
Read-only transactions that re-execute app code when the read set updates.

Automated end-to-end data management and storage with fault-tolerance, availability and consistency
Talk Outline

**Diamond System & Programming Model**
What does Diamond provide for reactive apps?

*Automated end-to-end data management and storage.*

**Diamond Guarantees & Implementation**
What does Diamond guarantee for reactive apps?

**Evaluation**
How does Diamond impact app complexity and performance?
Diamond ACID+R Guarantees

Atomicity - All or no updates to shared data in a read-write transaction complete.

Consistency - All accesses in a transaction (read-write or reactive) reflect a single, point-in-time view of shared data.

Isolation - All transactions reflect a serial execution order over shared data.

Durability - All updates in committed transactions are never lost.

Reactivity - All accesses in reactive transactions will eventually reflect the latest updates.
Diamond Isolation Levels

- Read-write Isolation Level
- Reactive Isolation Level

Stronger Guarantees

- Strict Serializability
- Serializable Snapshot

Better Performance

- Snapshot Isolation
- Serializable Snapshot

- Read Committed
- Read Committed
Diamond Implementation

Diamond Transaction Protocol

- Data-type Optimistic Concurrency Control
- Multi-versioned caching
- Data Push Notifications

Wide-area Optimizations

Take a look at the paper!
Talk Outline

Diamond System & Programming Model
What does Diamond provide for reactive apps?

Automated end-to-end data management and storage.

Diamond Guarantees & Implementation
What does Diamond guarantee for reactive apps?

Strong ACID+R transactional guarantees

Evaluation
How does Diamond impact app complexity and performance?
Evaluation Overview

• Does Diamond simplify reactive applications?
• How does Diamond perform compare to a hand coded implementation?
• Testbed: Google Compute Engine VMs (5 shards x 3 replicas)
• Workload: Retwis-based Twitter benchmark
Diamond reduces the complexity and improves the guarantees of reactive apps.

<table>
<thead>
<tr>
<th>Application</th>
<th>Original LoC</th>
<th>Diamond LoC</th>
<th>% Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-player Game</td>
<td>46</td>
<td>34</td>
<td>26%</td>
</tr>
<tr>
<td>Chat Room</td>
<td>335</td>
<td>225</td>
<td>33%</td>
</tr>
<tr>
<td>Scrabble clone</td>
<td>8729</td>
<td>7603</td>
<td>13%</td>
</tr>
<tr>
<td>Twitter clone</td>
<td>14,278</td>
<td>12,554</td>
<td>13%</td>
</tr>
</tbody>
</table>
Diamond reduces the complexity and improves the guarantees of reactive apps.

<table>
<thead>
<tr>
<th>Application</th>
<th>Original LoC</th>
<th>Diamond LoC</th>
<th>% Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-player Game</td>
<td>46</td>
<td>34</td>
<td>26%</td>
</tr>
<tr>
<td>Chat Room</td>
<td>335</td>
<td>225</td>
<td>33%</td>
</tr>
<tr>
<td>Scrabble clone</td>
<td>8729</td>
<td>7603</td>
<td>13%</td>
</tr>
<tr>
<td>Twitter clone</td>
<td>14,278</td>
<td>12,554</td>
<td>13%</td>
</tr>
</tbody>
</table>

No UI. Mostly sync code.
Diamond reduces the complexity and improves the guarantees of reactive apps.

<table>
<thead>
<tr>
<th>Application</th>
<th>Original LoC</th>
<th>Diamond LoC</th>
<th>% Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-player Game</td>
<td>46</td>
<td>34</td>
<td>26%</td>
</tr>
<tr>
<td>Chat Room</td>
<td>335</td>
<td>225</td>
<td>33%</td>
</tr>
<tr>
<td>Scrabble clone</td>
<td>8729</td>
<td>7603</td>
<td>13%</td>
</tr>
<tr>
<td>Twitter clone</td>
<td>14,278</td>
<td>12,554</td>
<td>13%</td>
</tr>
</tbody>
</table>

No UI. Mostly sync code.

Full UI. Complex app logic.
Diamond reduces the complexity and improves the guarantees of reactive apps.

<table>
<thead>
<tr>
<th>Application</th>
<th>Original LoC</th>
<th>Diamond LoC</th>
<th>% Saved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-player Game</td>
<td>46</td>
<td>34</td>
<td>26%</td>
</tr>
<tr>
<td>Chat Room</td>
<td>335</td>
<td>225</td>
<td>33%</td>
</tr>
<tr>
<td>Scrabble clone</td>
<td>8729</td>
<td>7603</td>
<td>13%</td>
</tr>
<tr>
<td>Twitter clone</td>
<td>14,278</td>
<td>12,554</td>
<td>13%</td>
</tr>
</tbody>
</table>

- No UI. Mostly sync code.
- Full UI. Complex app logic.

+durability
+reactivity
+consistency
+isolation
+durability
Diamond’s data management has low overhead.
Diamond’s data management has low overhead. 

-2.1%
Diamond’s data management has low overhead.

![Bar chart comparing throughput (txn/sec) between Redis and Diamond.](chart)

- Weak Consistency: Diamond has -2.1% lower throughput compared to Redis.
- Strong Consistency: Diamond has 24 more transactions per second compared to Redis.
Diamond’s data management has low overhead.

Throughput (txn/sec)

<table>
<thead>
<tr>
<th>Throughput</th>
<th>Redis</th>
<th>Diamond Read Committed</th>
<th>Diamond Strict Serializability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24K</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32K</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Weak Consistency: -2.1%
Strong Consistency: -48.5%
Linearizable Transactions: -48.5%
Summary

What does Diamond provide for reactive apps?
Automated end-to-end data management and storage.

What does Diamond guarantee for reactive apps?
Strong ACID+R transactional guarantees.

How does Diamond impact app complexity and performance?
Simplifies reactive apps with low overhead.

https://github.com/UWSysLab/diamond
Related Work

• Distributed Programming Frameworks
  Meteor, Parse, Firebase, Mjolnir, Mapjax, RethinkDB

• Client-side Programming Frameworks
  React, Angular, Blaze, ReactiveX

• Distributed Storage Systems
  Redis, MongoDB, Dropbox

• Notification/Pub-Sub/Streaming Services
  Thialfi, Apache Kafka, Amazon Kinesis