**Real-time system:** Một hệ thống máy tính có tính chính xác của hành vi của hệ thống không chỉ phụ thuộc vào kết quả logic của các tính toán mà còn về thời gian vật lý khi những kết quả được tạo ra.

* Hard: Microseconds – milliseconds, high risk.
* Soft: Milliseconds – seconds, low risk.
* Near: Seconds – minutes, low risk.

**Nhiều kiểu tương tác phổ biến:**

* **Request/response pattern**: the request and the response xảy ra trên cùng một kết nối. Service-side pattern can be half-async or full-async.
* **Request/acknowledge pattern**:
* **Publish/subscribe pattern**: Receive message without request
  + Firebase (cloud messaging):
    - Những lợi ích
      * Giảm mức sử dụng điện năng
      * Giảm tải máy chủ
      * Cải thiện khả năng mở rộng
      * Khớp nối lỏng lẻo
    - Hạn chế
      * Không đảm bảo giao hàng
      * Thứ tự và tính duy nhất của tin nhắn không được đảm bảo
      * Làm việc theo giả định khi địa vị của người tiêu dùng bị mờ đi

Multiple topics can attach to one subscription.

Multiple subscriptions can attach to one topic.

Subscription can have a filter to decide whether message will be sent or not.

* **One-way pattern**
  + Send request, but ignore response.
  + Especially useful for IoT applications.
  + Also known as “fire and forget”.
* **Stream pattern**

**Khả năng chịu lỗi(Fault tolerance) cho tầng bộ sưu tập**

• Một hoặc nhiều nút thu thập của chúng tôi có thể bị lỗi.

• Tin nhắn(message) có thể không được sao chép lại.

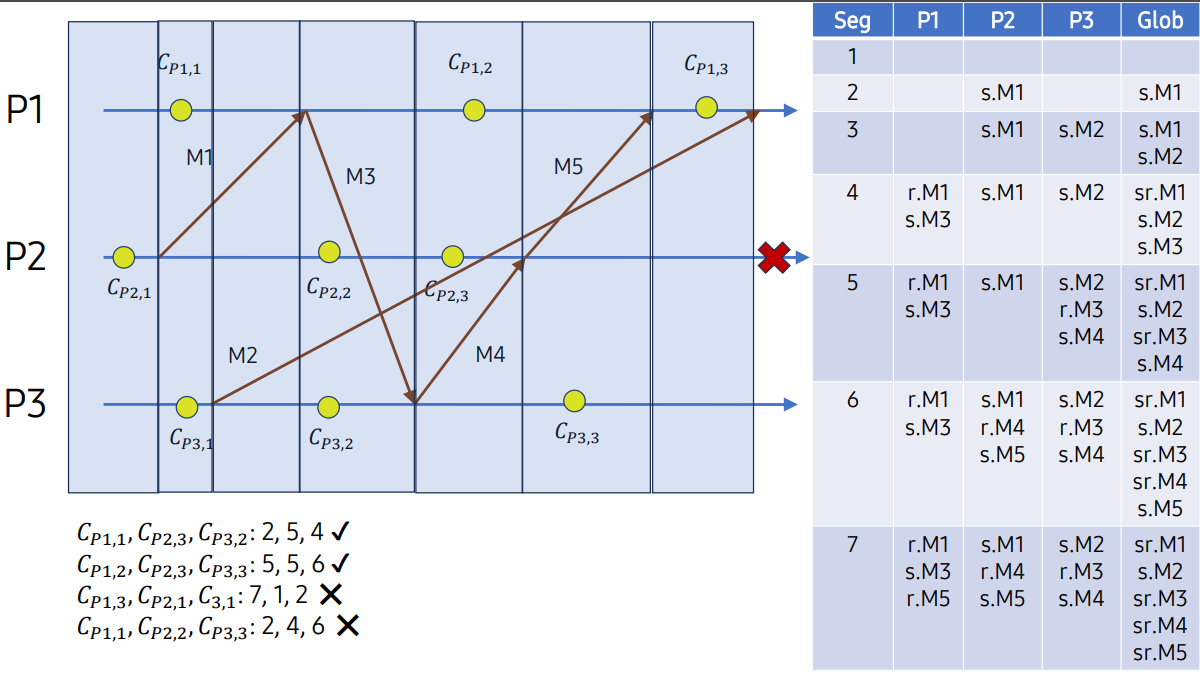
• Phương pháp tiếp cận: Checkpointing and logging

**Checkpointing**

Mục đích: Lưu trạng thái hệ thống để phục hồi.

Đặc trưng:

* Toàn cầu: Lưu toàn bộ trạng thái hệ thống.
* Khả năng mất dữ liệu: Khi khôi phục về trạng thái gần đây nhất, các tin nhắn được xử lý và tạo sau đó sẽ bị mất.



Không cần phải thất bại hoàn toàn mới có thể khôi phục trạng thái toàn cầu.

Mỗi tiến trình có thể có số lượng điểm kiểm tra khác nhau tại cột mốc khác nhau.

**Ghi nhật ký(Logging)**

• Mỗi tầng trong hệ thống ghi lại độc lập tất cả các tin nhắn mà nó nhận được và phát lại (phát lại) khi cần.

• Kỹ thuật: RBML, SBML & HML.

**Receiver-based message logging (RBML)**

• Phương pháp: Ghi tin nhắn đã nhận vào bộ lưu trữ ổn định trước khi thực hiện bất kỳ hành động nào đối với nó.

**Sender-based message logging (SBML)**

• Phương pháp: Viết nhận được thông báo tới bộ nhớ ổn định trước khi nó được gửi đi.

Một cách tiếp cận tốt hơn là “hybrid message logging” (HML).

• Quá trình phục hồi nhanh hơn và giảm tác động của việc thực thi không có lỗi.

**Message queue:**

• RabbitMQ

• ActiveMQ

• HornetQ

• NSQ

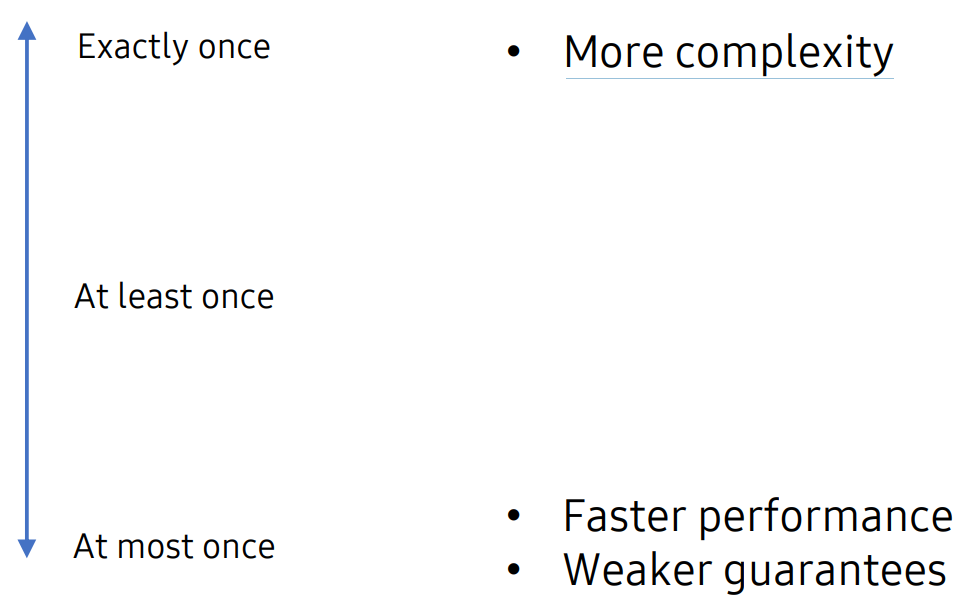
• ZeroMQ

• Apache Kafka

At most once: Tin nhắn có thể bị thất lạc, người tiêu dùng không thể đọc lại.

At least once: Tin nhắn không bao giờ bị thất lạc, người tiêu dùng có thể đọc lại.

Exactly once: Tin nhắn không bao giờ bị thất lạc, người tiêu dùng chỉ có thể đọc lại một lần.



**Bảo mật, khả năng chịu lỗi và các tình huống kinh doanh**

Bảo mật có thể được cải thiện bằng cách:

• Xác thực cấp độ

• Ủy quyền theo cấp độ

• Mã hóa kết nối

• Mã hóa lưu trữ

**Apache stream-processing technologies**

• Spark (streaming)

• Storm

• Flink

• Samza

**Key features**

Guarantees

• At-most-once: Message is dropped

• At-least-once: Message is safe, but can be reread

• Exactly-once: Ignore duplicates

**Giả định/ràng buộc(Assumptions/constraints)**

• One-pass: Không thể xử lý lại.

• Concept drift: Các mô hình dự đoán đã lỗi thời.

• Resource: Tốc độ dòng chảy làm căng hệ thống.

• Domain: Có yêu cầu kinh doanh.

**Một số công thức**

• Số lượng cửa sổ cần xem xét: k + kn - 1

• Số cửa sổ chứa hoàn toàn: kn − k + 1

• Số cửa sổ chồng lên nhau: 2(k − 1)

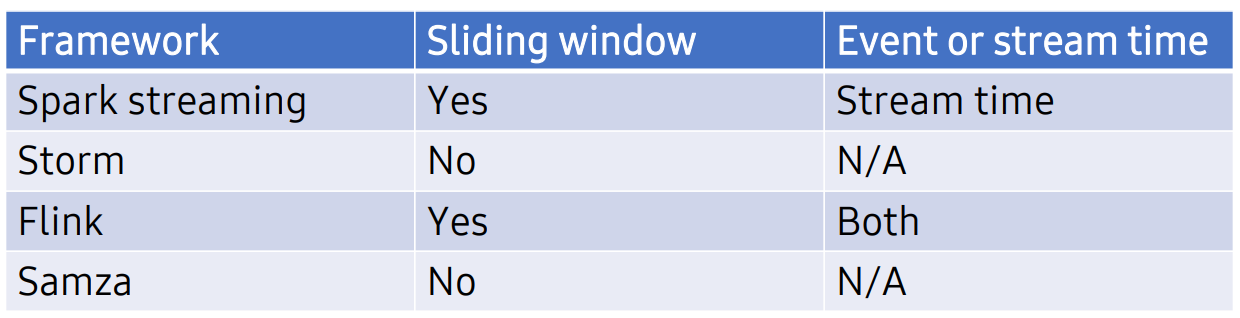
• k = w/s

• n = i/w

• w: Thời gian cửa sổ

• s: Thanh trượt thời gian

• i: Thời gian sự kiện/Thời gian ngắt quãng



**Nhào lộn: Chính sách số lượng(Tumbling: Quantity policy)**

• Chính sách trục xuất(Eviction policy): Khi đầy đủ.

• Chính sách kích hoạt(Trigger policy): Đếm số mục.

• 2 loại: Dựa trên số lượng và thời gian(Count-based & temporal).

Có xác suất(Probabilistic), một lần nữa!

• Dựa trên mẫu bit: Phát hiện các mẫu bit ở đầu giá trị nhị phân của mỗi phần tử.

LogLog, HyperLogLog, HyperLogLog++

• Dựa trên thống kê đơn hàng: Thống kê đơn hàng, chẳng hạn như các giá trị nhỏ nhất xuất hiện.

MinCount, Bar-Yossef

**HyperLogLog**

1. Lấy ID của phần tử.

2. Băm ID vào hàm băm.

3. Chuyển đổi thành chuỗi nhị phân, xác định giá trị thanh ghi nào

(bin) để cập nhật và giá trị cần cập nhật.

• Sử dụng n (độ chính xác) bit nhỏ nhất/có ý nghĩa nhất.

• n = log2 m

4. Xác định số số 0 đứng đầu 𝑣 bên phải chỉ số tại (3) và thêm 1 vào nó.

5. Cập nhật chỉ số tại (3) với giá trị v.

**Các loại trong bộ nhớ**

**• Được nhúng trong bộ nhớ/tối ưu hóa flash(Embedded in-memory/flash-optimized)**

- Toàn bộ hệ thống được gói gọn trong phần mềm của bạn.

- Nếu phần mềm hoạt động, cơ sở dữ liệu cũng hoạt động. Nếu nó bị hỏng, cơ sở dữ liệu bị hỏng.

- Công nghệ

• SQLite: Bộ nhớ cục bộ

• RocksDB: Lưu trữ nhanh khóa-giá trị

• LevelDB: Ánh xạ khóa-giá trị theo thứ tự

• LMDB: Dành cho khối lượng công việc đọc nhiều

• Perset: Hệ thống cơ sở dữ liệu nhúng cho Java & .NET

• IndexedDB: Cơ sở dữ liệu được lưu trữ bên trong trình duyệt

**• Bộ nhớ đệm(Caching system)**

- Lưu trữ dữ liệu trong bộ nhớ.

- Không có tùy chọn lưu trữ bên ngoài RAM.

- 5 chiến lược:

**• Read-through**

**• Refresh-ahead**

• Thường xuyên làm mới dữ liệu được truy cập trong bộ nhớ đệm.

• Trừ khi dữ liệu được yêu cầu ban đầu, dữ liệu sẽ luôn được lấy từ bộ đệm.

• Có thể không phù hợp với bộ nhớ đệm lớn.

**• Write-through**

• Hệ thống bộ nhớ đệm sẽ ghi dữ liệu.

• Bộ nhớ đệm được cập nhật trong khi ghi dữ liệu.

Tuy nhiên,

• độ trễ ghi tăng lên.

• bộ nhớ đệm có thể không biết việc ghi dữ liệu có thành công hay không.

**• Write-around**

• Cập nhật dữ liệu vào bộ lưu trữ thông qua một quy trình phức tạp hơn.

• Bộ nhận cập nhật tại bộ lưu trữ sẽ gửi dữ liệu vào bộ nhớ đệm.

• Bộ đệm thụ động.

**• Write-back**

• Nâng cấp khả năng ghi.

• Trong khi cập nhật bộ đệm, bộ lưu trữ sẽ nhận dữ liệu mới ở chế độ nền.

• Giảm độ trễ khi viết.

• Tăng nguy cơ mất dữ liệu.

- **Technologies**

• Memcached

• EHCache

• Hazelcast

• Redis

**• Cơ sở dữ liệu trong bộ nhớ/Lưới dữ liệu trong bộ nhớ(In-memory database (IMDB))**

**- Đôi khi được gọi**

• Hệ thống quản lý cơ sở dữ liệu trong bộ nhớ

•In-memory data grids (IMDG), nhưng có sự khác biệt nhỏ

**- Technologies**

• SingleStore (MemSQL)

• VoltDB

• Aerospike

• Apache Geode

• Couchbase

• Apache Ignite

• Hazelcast

• Inifispan