INTELLIGENCE · BUSINESS INTELLIGENCE

Real-Time Intelligence and Streaming Data

Some business decisions lose their value within minutes. This article explains what real-time intelligence means, how streaming data systems work, where speed pays off, and the risks of reacting too quickly.

ModQu research article · 1,425 words · 15 references (APA 7th edition)

Traditional business intelligence looks backwards. Data are collected during the day, loaded into a warehouse overnight and reported the next morning. For many decisions, such as quarterly planning or annual budgeting, that rhythm is fine. For others it is far too slow. A fraudulent card transaction must be stopped before it is approved; a customer about to abandon an online purchase must be engaged while still on the website; a fault on a production line must be caught before hundreds of defective units are made. **Real-time intelligence** aims to deliver knowledge quickly enough to act while action still matters. This article explains what real-time intelligence involves, the technology behind it, where it pays off and where it can mislead.

What "real time" means

In business intelligence, "real time" rarely means instantaneous. It means fast enough for the decision at hand. For fraud detection that may be milliseconds; for managing staff in a call centre, minutes; for adjusting a marketing campaign, hours. The useful question is not "how fast can we go?" but "how quickly does the value of acting on this information decline?"

The delay between an event and the action taken in response has several components, shown in Figure 1. **Data latency** is the time to capture and store the event. **Analysis latency** is the time to process the data and detect something meaningful. **Decision latency** is the time for a person or system to receive the information and act on it.

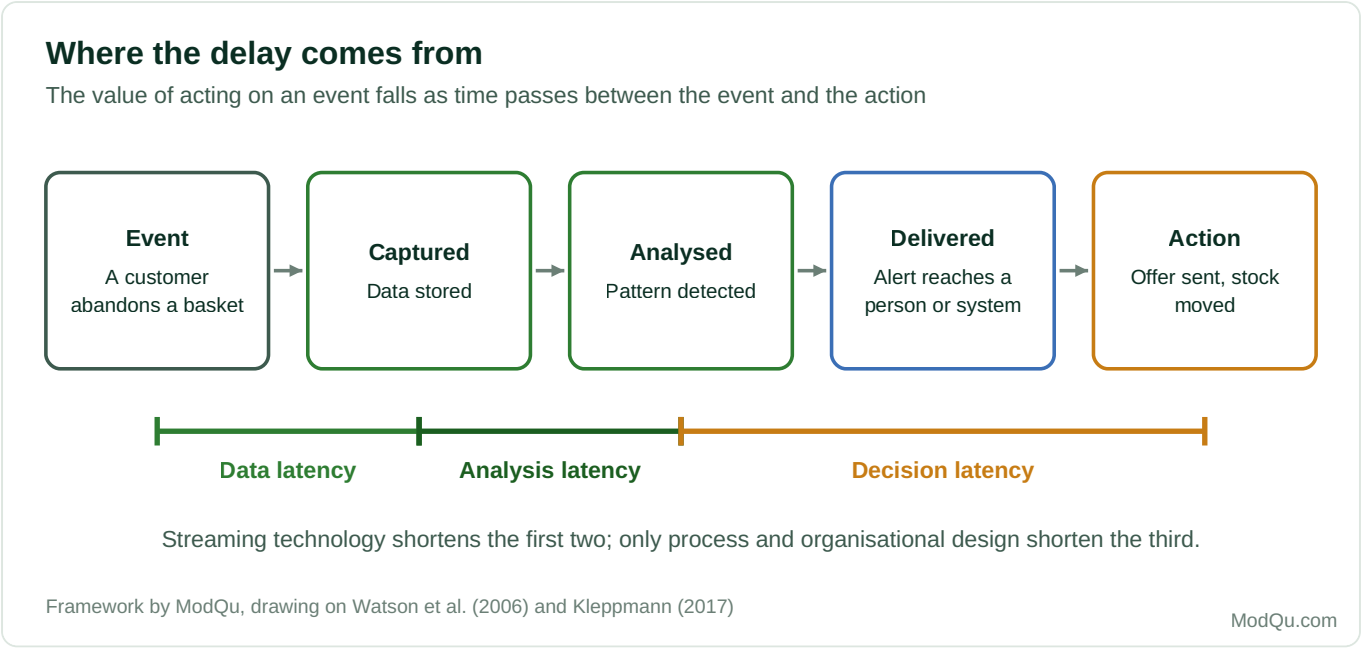


Figure 1. Where the delay comes from. Faster data systems reduce only part of the total delay. Framework by ModQu, drawing on Watson et al. (2006) and Kleppmann (2017).

Organisations often invest heavily in reducing data and analysis latency, only to find that alerts sit unread or that no one has authority to act on them. Real-time data without real-time decision processes adds cost but little value.

How streaming systems work

Conventional analytics processes data in **batches**: a large set of records is collected and processed together at intervals. Dean and Ghemawat (2008) described MapReduce, a model for processing very large batches across many machines that underpinned the first generation of big data systems. **Stream processing** instead handles each event, or small groups of events, as it arrives.

Babcock et al. (2002) set out the core problem of data stream systems: data arrive continuously, possibly at high rates, and must be processed with limited memory, often in a single pass, producing answers that are continuously updated. Stonebraker et al. (2005) listed requirements for real-time stream processing, including keeping data moving without storing it first, handling late and out-of-order data, and producing predictable results.

Several building blocks are now common:

- **Event logs and messaging.** Kreps et al. (2011) described Kafka, a system developed at LinkedIn to collect and distribute large volumes of event data, such as page views and clicks, to many consuming applications. Such logs act as a central nervous system through which events flow from where they occur to where they are analysed.
- **Stream processing engines.** Zaharia et al. (2016) describe Apache Spark, which supports both batch and streaming computation within one engine. Akidau et al. (2015) proposed the dataflow model, which addresses a central difficulty: events often arrive late or out of order, so a system must decide when it has seen enough data to produce a result, trading off completeness, speed and cost.

- **Complex event processing.** Luckham (2002) introduced methods for detecting meaningful patterns across many simple events, such as a sequence of small transactions in different cities within an hour that together suggest fraud. Cugola and Margara (2012) review the range of systems developed to process such flows of information.

Kleppmann (2017) emphasises that these systems involve trade-offs between consistency, availability, latency and complexity. Streaming systems are harder to build and operate than batch systems, and they should be used where speed justifies that cost.

Where speed pays off

Real-time intelligence is most valuable where the value of information decays quickly and where action can follow immediately. Common examples include:

- **Fraud and security.** Card payments, insurance claims and network intrusions can be scored as they occur and blocked or flagged before harm is done.
- **Operations.** Sensors on equipment, vehicles and production lines reveal faults and bottlenecks while they can still be fixed.
- **Customer interactions.** Websites and call centres can respond to what a customer is doing now, recommending products or routing calls to the right agent.
- **Supply and logistics.** Real-time tracking of shipments and inventory lets companies reroute deliveries and rebalance stock.
- **Pricing.** Airlines, hotels and online retailers adjust prices in response to demand.

Watson et al. (2006) describe how Continental Airlines used real-time business intelligence to manage flight operations and customer service, giving staff current information about delayed flights, connecting passengers and high-value customers. The value came not only from faster data but from changes in how staff were expected to use it, such as rebooking passengers before they reached the airport.

Endsley (1995) describes situation awareness as perceiving the current state of the environment, understanding its meaning and projecting its future state. Real-time intelligence supports all three for operational managers, but only if displays are designed to highlight what needs attention rather than overwhelming users with constant updates.

The risks of speed

Reacting to noise

Every measure fluctuates. Shewhart (1931) distinguished routine variation from variation that signals a real change. The more frequently a measure is updated, the more of its movement is noise. Managers watching a real-time dashboard may react to random fluctuations, making changes that do nothing or make matters worse. Real-time displays should show expected ranges of variation and alert only on changes that exceed them.

Models that drift

Real-time systems often apply predictive models automatically. Gama et al. (2014) review the problem of **concept drift**: the relationships a model has learned change over time as customer behaviour, products or the environment change. A fraud model trained last year may miss new fraud patterns or flag legitimate behaviour. Streaming systems need continuous monitoring of model performance and mechanisms to adapt.

Lazer et al. (2014) showed how a system that estimated influenza activity from search queries drifted badly over time as search behaviour changed. The lesson applies to any automated real-time indicator: it must be validated against independent measures and recalibrated.

Incomplete data

Streaming results are based on the data received so far. Late-arriving events, network failures and duplicated messages can make early results wrong (Akidau et al., 2015). Users need to know how complete and reliable a real-time figure is, and decisions with large consequences may need to wait for confirmation.

Cost and complexity

Real-time architectures are more expensive to build and maintain than batch systems. Chen et al. (2012) note that the growth of mobile and sensor data has made real-time analysis increasingly important, but many organisations find that hourly or daily updates meet most of their needs at a fraction of the cost.

An example

A regional electricity distributor installs sensors that report the condition of transformers every few seconds. At first, the readings feed a dashboard in the control room, and operators are swamped by alerts for minor temperature changes. The company then redesigns the system. Readings are compared with each transformer's own normal range, adjusted for weather and load; only sustained departures trigger an alert; and each alert type is linked to a defined response, from scheduling an inspection to dispatching a crew immediately. Alerts fall by more than half, and operators begin to trust and act on the ones that remain.

Designing for real-time intelligence

1. **Start from decisions.** Identify which decisions lose value with delay, and by how much.
2. **Match speed to need.** Use streaming where it pays; keep batch processing elsewhere.
3. **Design the response.** Decide in advance who or what will act on each alert, with what authority, and how quickly.
4. **Automate where appropriate.** Where decisions are routine and well understood, such as blocking a clearly fraudulent transaction, automated action removes decision latency altogether.
5. **Filter noise.** Alert on significant changes, not every movement.
6. **Monitor and recalibrate.** Track the accuracy of real-time models and alerts, and adjust as conditions change.

Conclusion

Real-time intelligence lets organisations act while action still matters. Streaming technologies have made it practical to capture and analyse events within seconds, but speed alone creates little value. The benefits come from identifying decisions that truly depend on timeliness, designing processes to act on information quickly and filtering signal from noise. Used selectively and with discipline, real-time intelligence is a powerful complement to the slower, deeper analysis of traditional business intelligence.

References

Akidau, T., Bradshaw, R., Chambers, C., Chernyak, S., Fernández-Moctezuma, R. J., Lax, R., McVeety, S., Mills, D., Perry, F., Schmidt, E., & Whittle, S. (2015). The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, *8*(12), 1792–1803. https://doi.org/10.14778/2824032.2824076

Babcock, B., Babu, S., Datar, M., Motwani, R., & Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (pp. 1–16). ACM. https://doi.org/10.1145/543613.543615

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly*, *36*(4), 1165–1188. https://doi.org/10.2307/41703503

Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, *44*(3), Article 15. https://doi.org/10.1145/2187671.2187677

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. https://doi.org/10.1145/1327452.1327492

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors*, *37*(1), 32–64. https://doi.org/10.1518/001872095779049543

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46*(4), Article 44. https://doi.org/10.1145/2523813

Kleppmann, M. (2017). *Designing data-intensive applications*. O'Reilly Media.

Kreps, J., Narkhede, N., & Rao, J. (2011). Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB Workshop*. ACM.

Lazer, D., Kennedy, R., King, G., & Vespignani, A. (2014). The parable of Google Flu: Traps in big data analysis. *Science*, *343*(6176), 1203–1205. https://doi.org/10.1126/science.1248506

Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Addison-Wesley.

Shewhart, W. A. (1931). *Economic control of quality of manufactured product*. Van Nostrand.

Stonebraker, M., Çetintemel, U., & Zdonik, S. (2005). The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, *34*(4), 42–47. https://doi.org/10.1145/1107499.1107504

Watson, H. J., Wixom, B. H., Hoffer, J. A., Anderson-Lehman, R., & Reynolds, A. M. (2006). Real-time business intelligence: Best practices at Continental Airlines. *Information Systems Management*, *23*(1), 7–18. https://doi.org/10.1201/1078.10580530/45769.23.1.20061201/91768.2

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, *59*(11), 56–65. https://doi.org/10.1145/2934664