Temporal graphs capture the development of relationships within data throughout time. This model fits naturally within a streaming architecture, where new events can be inserted directly into the graph upon arrival from a data source, being compared to related entities or historical state. However, the vast majority of graph processing systems only consider traditional graph analysis on static data, with some outliers supporting batched updating and temporal analysis across graph snapshots. With this in mind, our recent work has been looking at defining a temporal graph model which can be updated via event streams and investigating the challenges of distribution and graph maintenance. Some notable challenges include partitioning a graph built from a stream, with the additional complexity of managing trade-offs between structural locality (proximity to neighbours) and temporal locality (proximity to an entities history). Synchronising graph state across the cluster and handling out-of-order updates, without a central ground truth limiting scalability. Managing memory constraints and performing analysis in parallel with ongoing update ingestion.

To address these challenges, we introduce Raphtory, a system which maintains temporal graphs over a distributed set of partitions, ingesting and processing parallel updates in near real-time. Raphtory's core components consist of Graph Routers and Graph Partition Managers. Graph Routers attach to a given input stream and convert raw data into graph updates, forwarding this to the Graph Partition Manager handling the affected entity. Graph Partition Managers contain a partition of the overall graph, inserting updates into the histories of affected entities at the correct chronological position. This removes the need for centralised synchronisation, as commands may be executed in any given arrival order whilst resulting in the same history. To deal with memory constraints, Partition Managers both compress older history and set an absolute threshold for memory usage. If this threshold is met a cut-off point is established, requiring all updates prior to this time to be transferred to offline storage. Once established and ingesting the selected input, analysis on the graph is permitted via Live Analysis Managers. These connect to the cluster, broadcasting requests to all Partition Managers who execute the algorithm. Analysis may be completed on the live graph, or any point back through its history, with Raphtory handling the retrieval of data which has been pushed to disk. Additionally, multiple Analysis Managers may operate concurrently on the graph with previously unseen algorithms compiled at run-time, thus allowing modification of ongoing analysis without re-ingesting the data.

Raphtory is an ongoing project but has a working version available with all of the above components containerised for ease of installation and reproducibility of tests. Much work has also gone into making it simple for users to ingest their own data sources, create custom routers and perform their desired analysis. The current goals of the project are to expand upon initial testing, including several real world uses cases, and to extend the systems API to better enable true temporal analysis. The proposed talk will, however, focus predominantly on the developed components to gather feedback on these, with several areas of expansion introduced at the end for discussion with those interested.