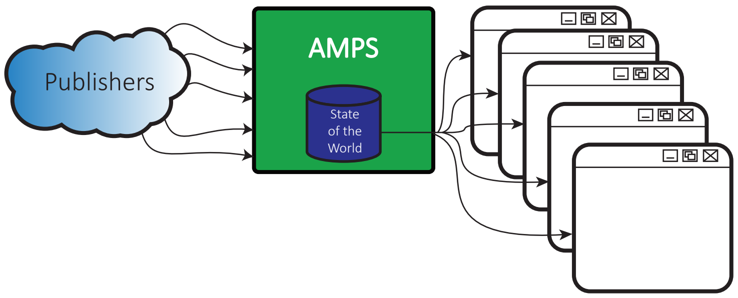
ADVANCED MESSAGING DATABASE

APPLICATION

Taking in the View: AMPS as View Server

Easy or High-Performance? Pick Both!

AMPS is often used as a persistent, high-performance back end for GUI clients. One AMPS server can provide data for a large number of GUI clients, and the state-of-the-world database, replicated topics and content filtering make sure that clients recieve only the data they need. This reduces network traffic and makes it easier to write efficient clients.



AMPS handles the difficult parts of creating the server: managing client subscriptions, reducing chatty network traffic, filtering data for relevance, and informing clients when data is no longer relevant.

Simplicity is Efficiency

AMPS is engineered for efficient message delivery. Part of the 60East philosophy is that simplicity is often the shortest path to efficiency. 60East's engineering makes it simple to build a view server. AMPS provides:

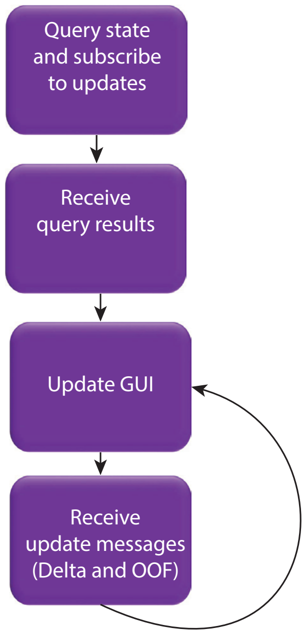
State-of-the-World (SOW) databases keep track of the current values of data. In a view server, clients use the SOW to load current values and to retrieve reference data. At any time, your program can easily get current state, just as though it had tracked the state changes in every message up to that point.

Content filtering allows you to retrieve and subscribe to only data that needs to be displayed. This reduces network traffic, improves end-to-end latency, and makes it easier to build an efficient system since you know your program only receives relevant messages. For example, you can choose to subscribe to only updates that affect the current user or orders that haven't been assigned. Each instance of the client can provide a different filter -- meaning that your program only gets the messages relevant for that instance of the program.

Atomic query and subscribe functionality allows you to query for historical state and subscribe to updates in a single, atomic operation. AMPS guarantees that no messages are lost.

Out-of-focus notification makes it easy for your program to display only current, relevant data. When data no longer matches the content filter, AMPS lets you know. For example, when an order is assigned to a different user, AMPS sends a dedicated out-of-focus notification to tell your program to stop showing that order.

Delta messages reduce network traffic and client-side processing even further, and help you to focus on the important data. With delta messages, AMPS provides only updated fields — the changes, or deltas— rather than complete messages. Your application does not have to compare before-and-after state to know what's changed. Every field in the message is an update that your program will act on; you don't need to waste cycles comparing before-and-after state.


Query state and subscribe to updates
Receive query results
Update GUI
Receive update messages (Delta and OOF)

Putting it All Together

To create a view server, you simply request the current state of the world, filtered for relevance. You request out-of-focus notification and subscribe to delta updates. AMPS returns complete messages for the current state of the world, and then notifies you of relevant changes. While the exact statements you use depend on the programming language you choose, the overall program flow is the same.

Your program connects to AMPS, then requests the current state of the world. AMPS delivers this to your program as a series of messages that contain the current state. You use these messages to update the GUI with the data your client needs to see. As your program runs, it continues to receive updates. Each update is either a new message, a set of changed fields for an existing message, or a notification that a message is no longer relevant. Each message is actionable, and the program updates the GUI when each message arrives.

To learn more about how AMPS makes it simple and easy to build a view server, contact us at info@crankuptheamps.com. Or visit our website today to download an evaluation copy and CRANK UP THE AMPS!