



LIVE CODING



Writing Highly Performant PI Web API Applications

Presented by **Jim Bazis & Max Drexel**



Introduction



Max Drexel

mdrexel@osisoft.com

Software Developer

PI Web API Team



Jim Bazis

jbazis@osisoft.com

Team Lead

PI Web API Team

Agenda

- Goals
- Streamsets
- Batch Requests
- Advanced Batch
- Channels
- Stream Updates





Overview

Goals

- Understand common sources of poor performance
- Recognize them when they occur
- Know your options to mitigate them

What do we mean by “Poor Performance”?

- Pages load too slowly for end users
- Too many users causes quality of service to degrade
- Can't acquire data fast enough for application to be useful
- *And many more...*

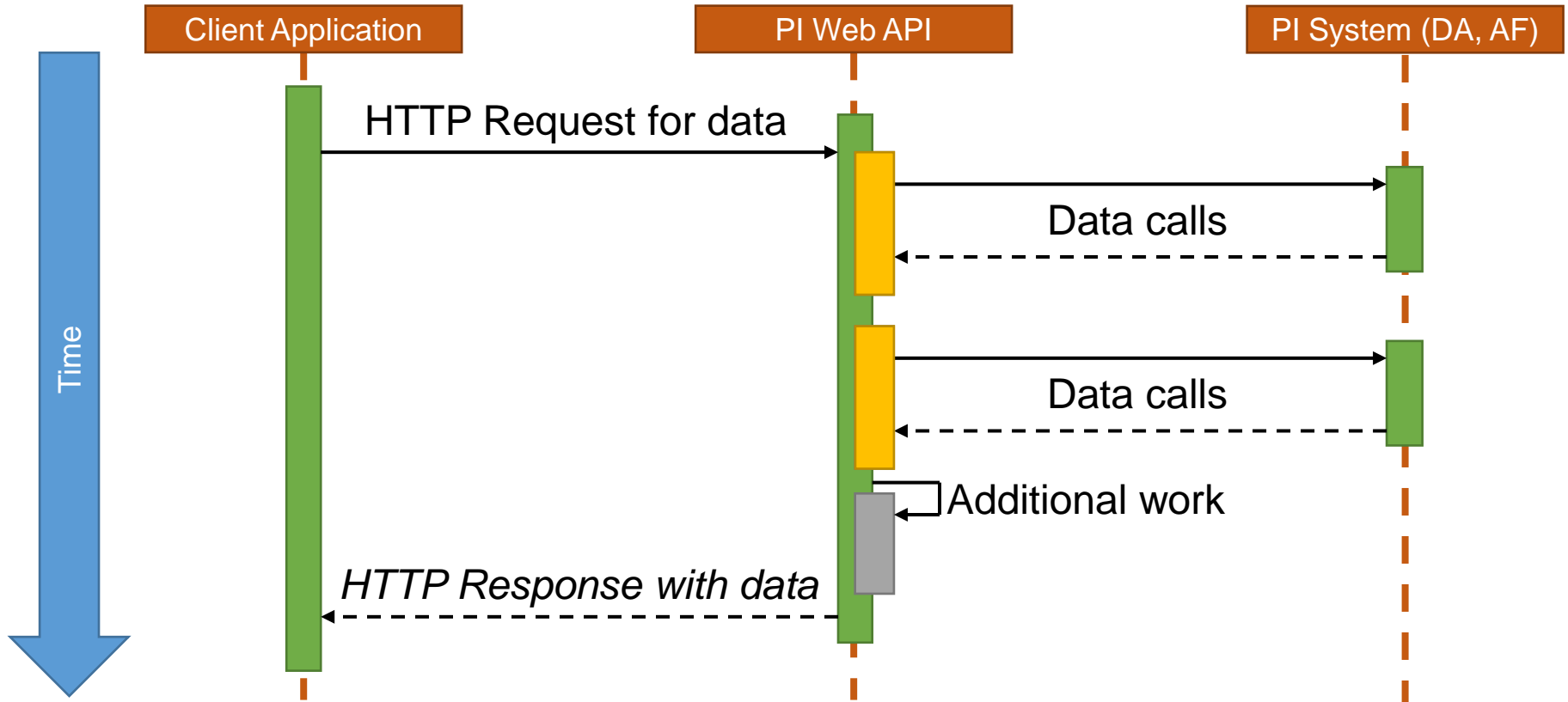


LIVE CODING

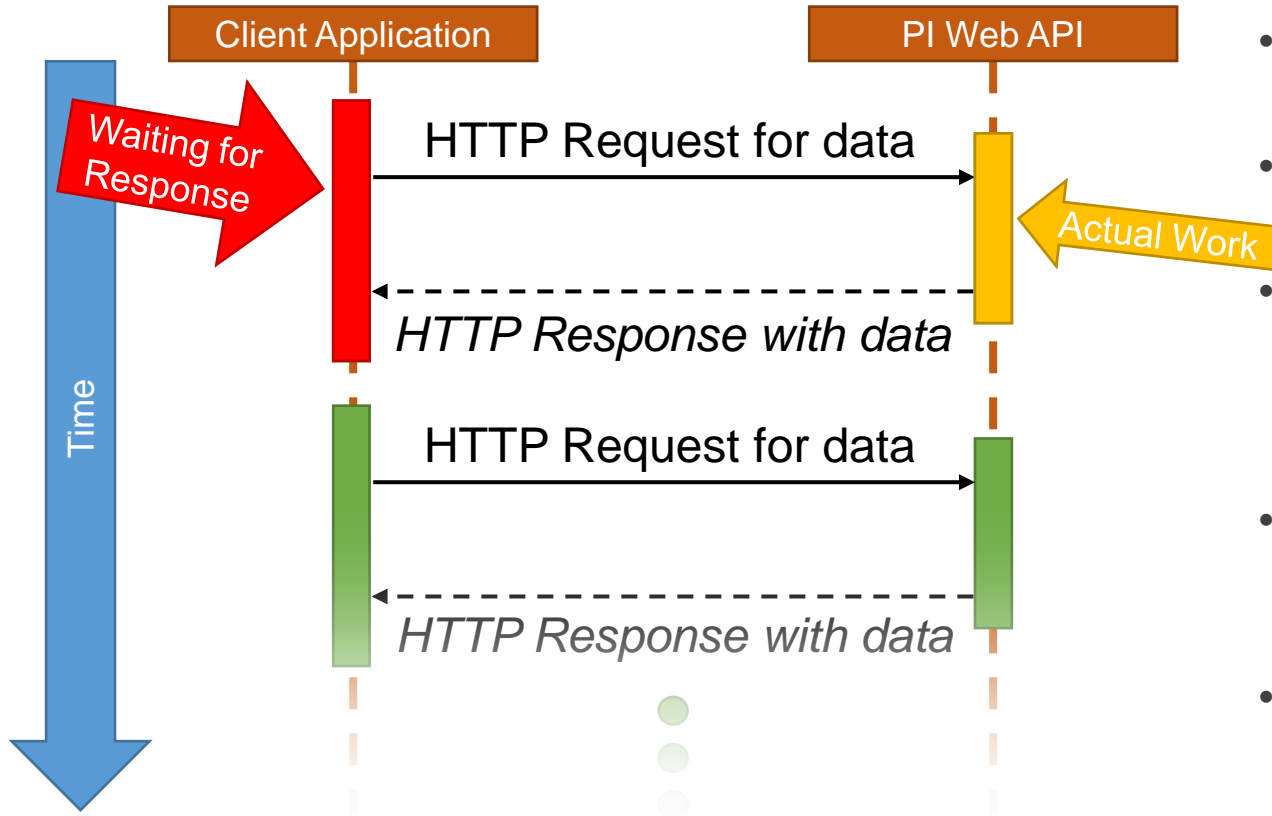


Example Application

PI Web API HTTP request lifecycle



What are the constraints of this lifecycle?



- Every request incurs a performance penalty
- Network latency & bandwidth
- Ethernet, TCP/IP, and TLS: reduces throughput to 87.7% under ideal conditions
- Calls to other services (identity provider, AF server, etc.)
- We don't control these

What *do* we have control over?

- The resources we interact with
- The way we interact with them
 - Which endpoints we call
 - How frequently we call them
 - What communication mechanism we use

Optimize!

- Requests can't escape latency; make fewer requests
- Bandwidth is limited; use less of it
- Server has finite resources; use them more efficiently



Areas of Improvement

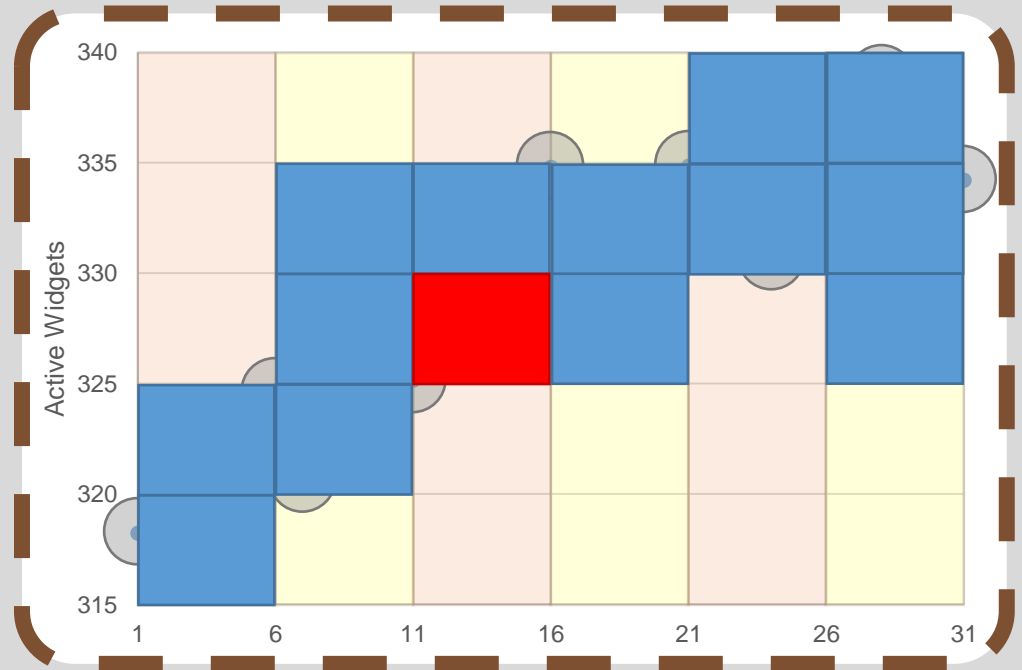
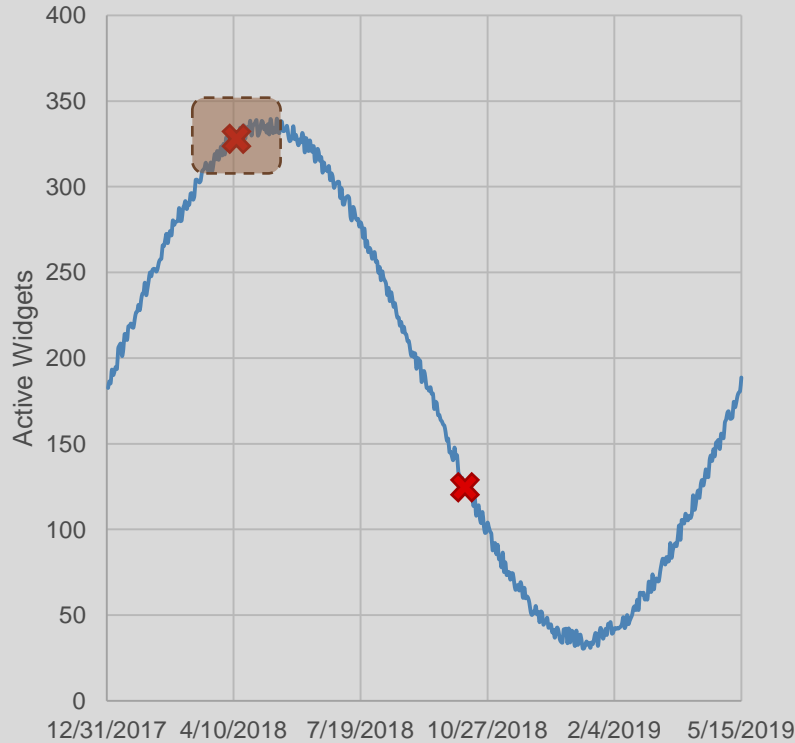
Endpoints & Resources

PI Web API Endpoints

- Some endpoints are designed to improve performance
 - *What* are they?
 - *How* do I use them?
 - *Which* one is appropriate for my use case?

Plot Values

What is plot data?





LIVE CODING

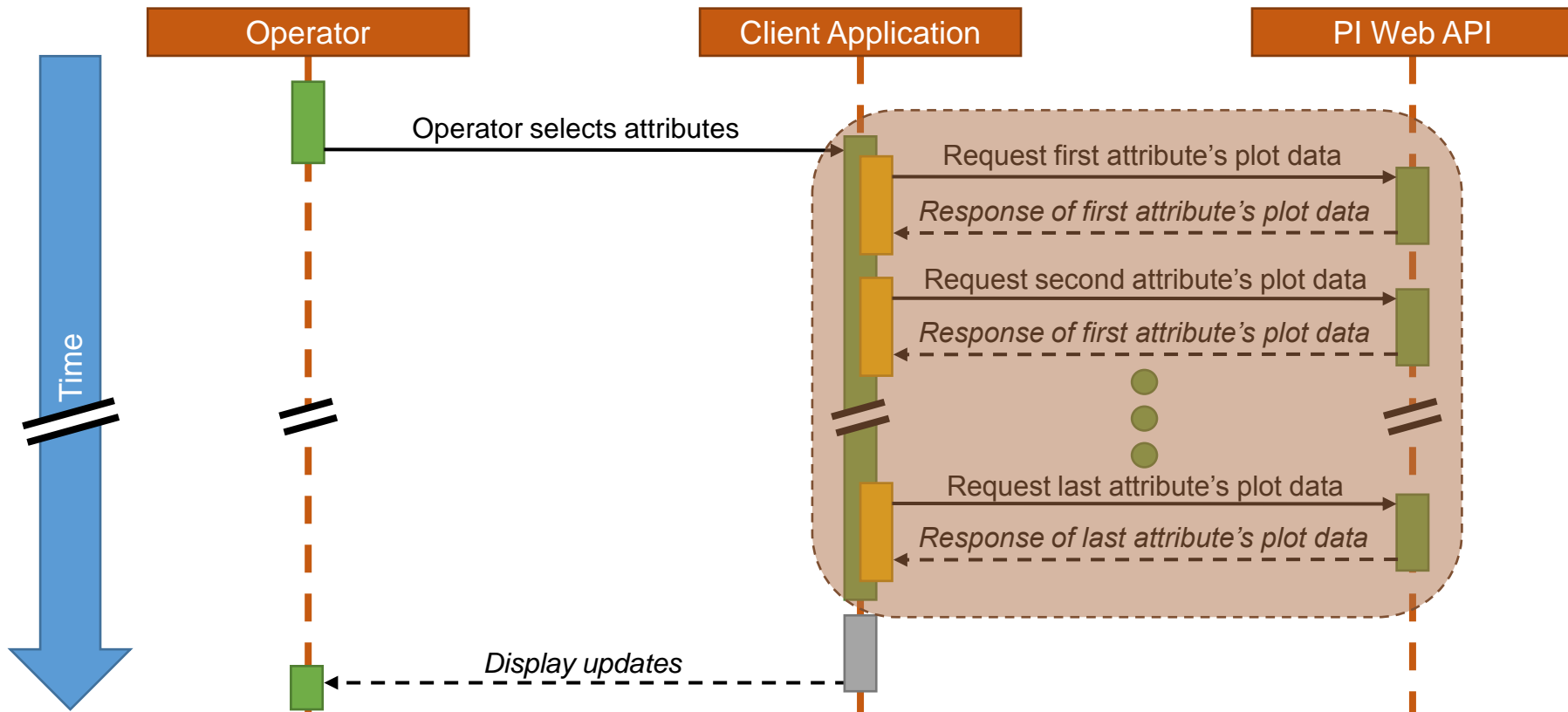


INTRODUCTORY

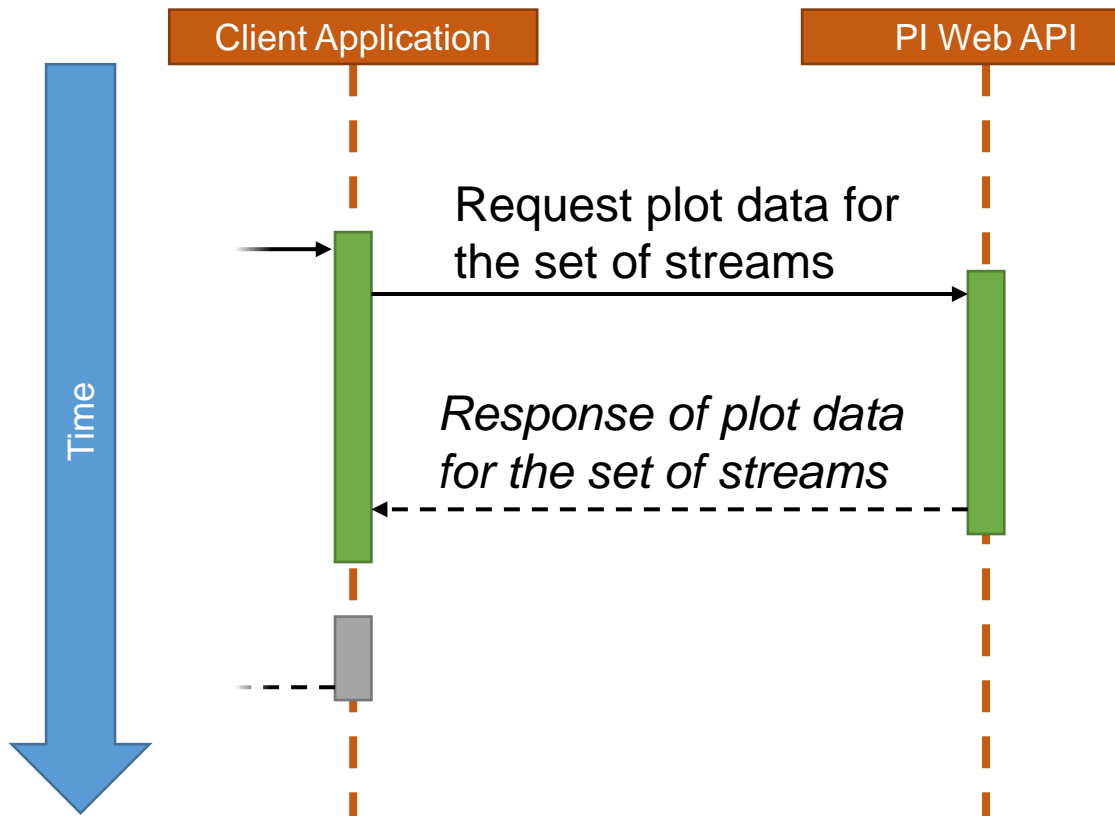
Plot Values

(Continued)

Example application's current behavior



PI Web API features: Stream Sets



- Use a “Stream Set” request
- Reduced to a single HTTP request
- Many round-trips removed
- PI Web API can optimize backend calls for even better performance



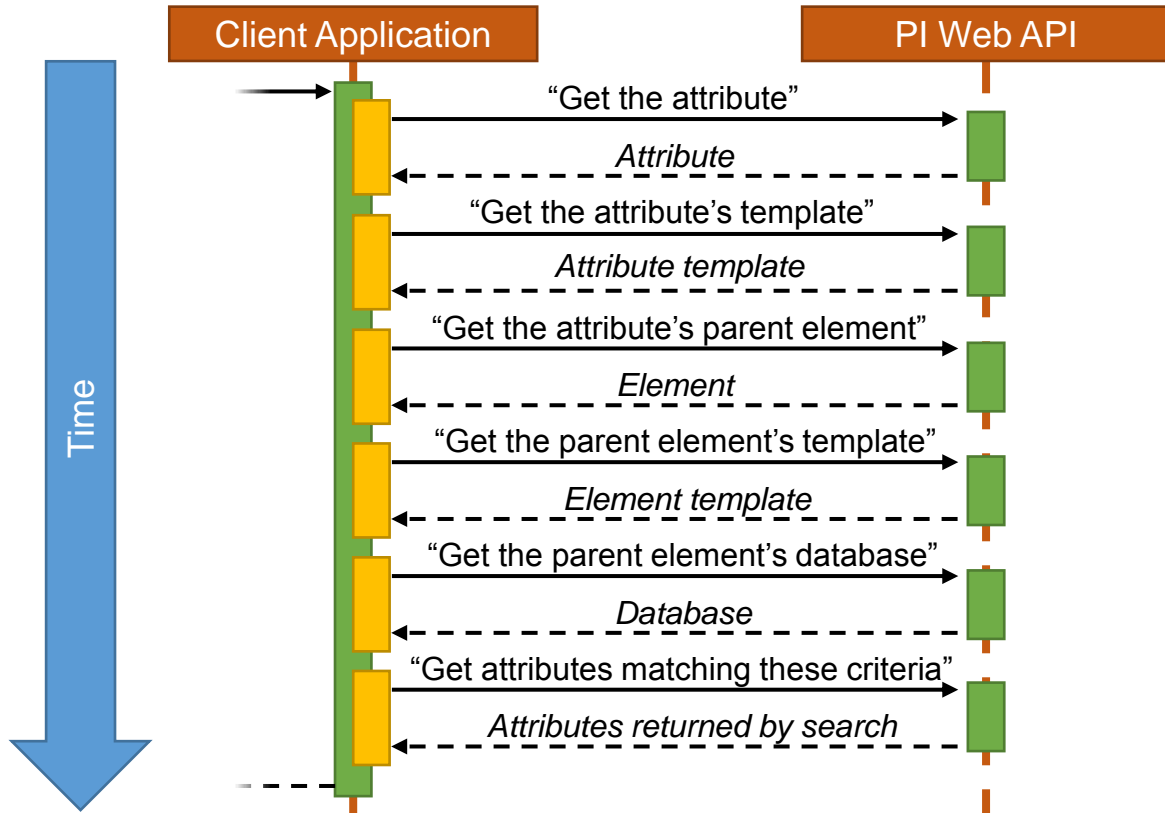
LIVE CODING



INTRODUCTORY

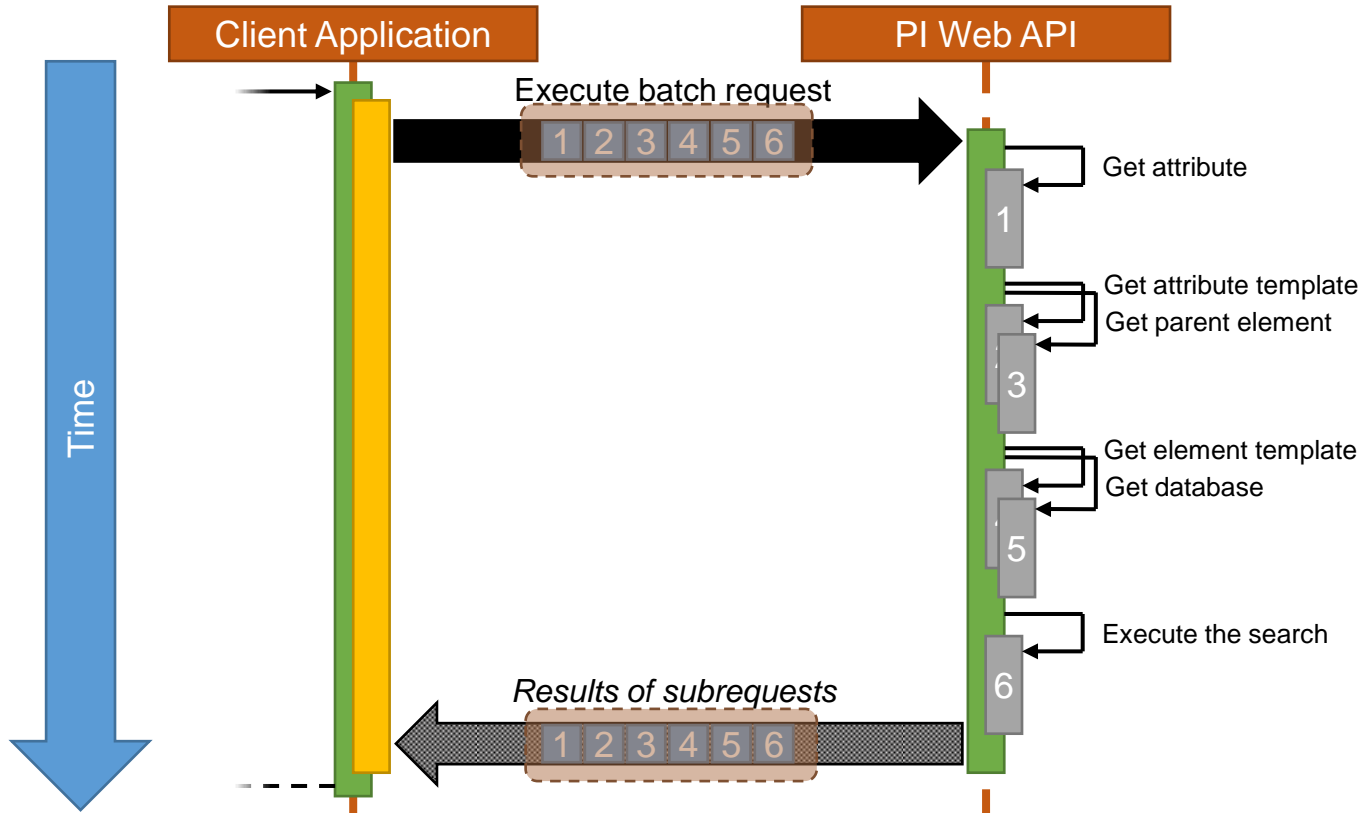
Stream Sets

Same problem, different example



- Problem statement: "Given an data point on a car, get the same data point on the other cars"
- Formalized as: "Given an attribute, get all attributes using the same attribute template"
- Robust implementation ends up taking 6 requests

Modified to use advanced PI Web API features



- Logic can be bundled into a single Batch request
- PI Web API can parallelize non-dependent requests
- Batch subrequests are executed without needing to traverse the network
- Results of the subrequests are sent as a single response
- Now we only need one round-trip: five removed! Free performance!




LIVE CODING



INTERMEDIATE

Batch



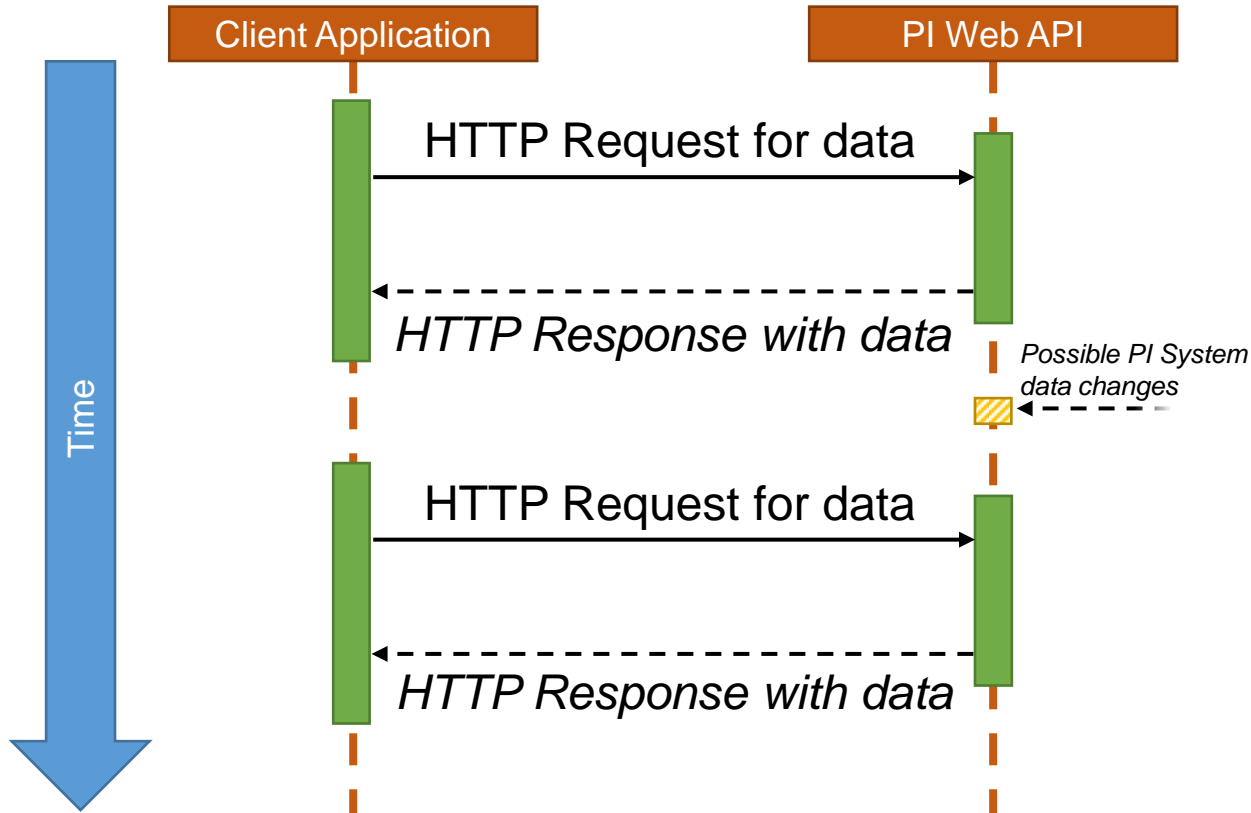
Areas of Improvement

Communication Mechanisms

Communication mechanisms

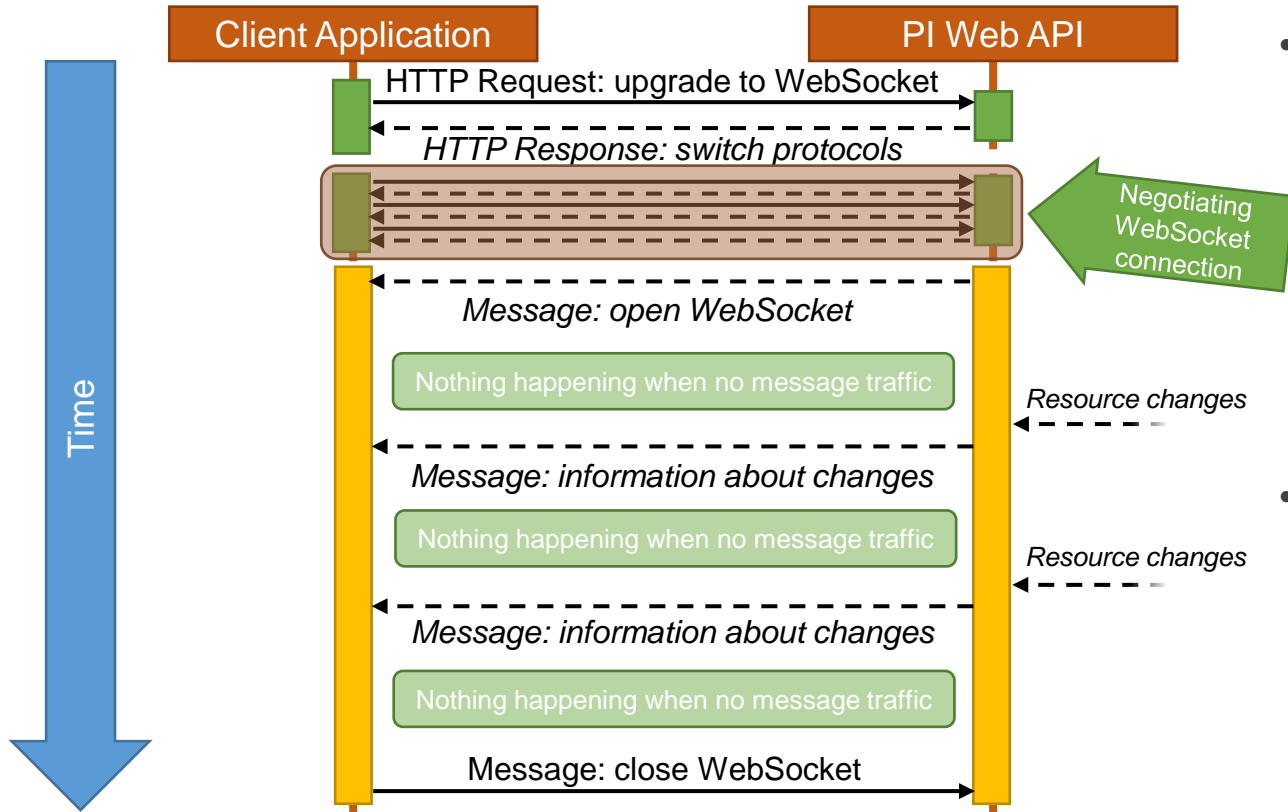
- What are the options?
- Which one should I use?
- How do they impact my application?

HTTP Requests



- Pros:
 - Easy to use
 - Widely supported
 - Can make use of existing infrastructure (load balancers, analytics, caching, etc.)
 - Low hardware overhead
 - Low software overhead
- Cons:
 - Need to continually issue requests to find out about changes (polling)

WebSockets (using the Channels feature)



• Pros:

- Get informed of changes as they occur: no polling needed
- Lower latency
- Less protocol overhead: never need more than 14 bytes per frame (vs. HTTP headers – still suffer from TCP/TLS/etc.)
- Asynchronous model – not wasting hardware or network resources

• Cons:

- Need client support
- **Underlying TCP connection still has network traffic**
- *PI Web API specific*: Does not support Claims Based Authentication



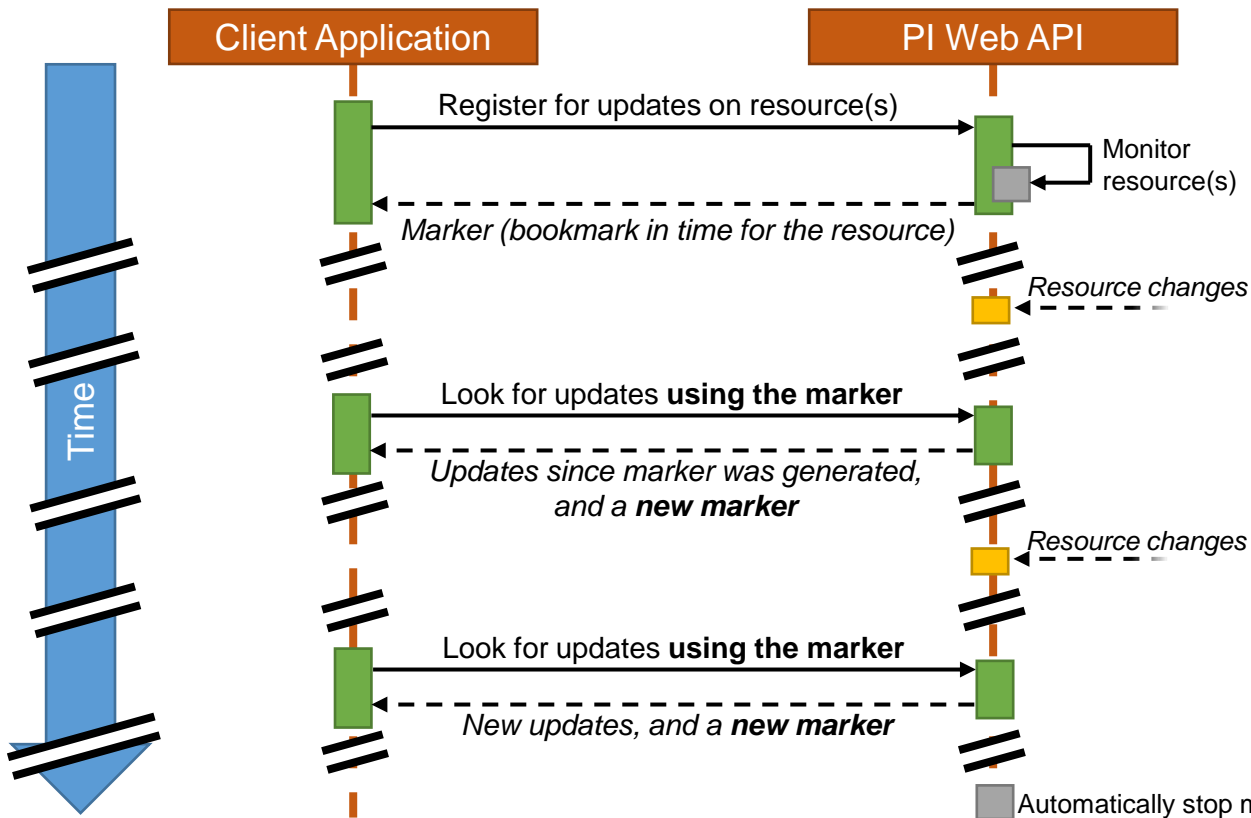
LIVE CODING



ADVANCED

Channels

Stream Updates (CTP)



- Registers the stream or streamset to be monitored for changes
- Every time you request the updates, you get the changes since the time you registered and a new link to use next time
- Pros:
 - Operates over HTTP – get all the benefits of normal HTTP requests (infrastructure, library support, etc.)
 - Response sizes are much smaller than polling (only getting changes)
 - Uses less server & network resources than polling
 - Works with Claims Based Authentication
- Cons:
 - Client application needs to actively check for changes (not as easy as Channels)
 - Registrations are per Web API instance (need sticky sessions)



LIVE CODING



ADVANCED

Stream Updates

PI Web API AFSearch Functionality

- Introduced as part of PI Web API 2017 R2
- Uses AF Search syntax
- As of PI Web API 2018, the following search types are supported:
 - Analyses
 - Analysis Templates
 - Attributes
 - Elements
 - Event Frames
 - Notification Rules
 - Notification Rule Templates
- Much better performance, especially for use cases where users frequently re-execute searches
- Uses fewer resources across the PI System

PI Web API Configuration Tweaks

Name	Description	Use Case
AFCacheRefreshHoldoffTime	How long (in milliseconds) the PI Web API should defer checking PI/AF for changes when a request requires fresh (non-cached) data.	When a very high volume of PI System data read requests are expected (ex. multiple Recorded, Plot, or interpolated calls are outstanding at any given time), higher values give better performance. <i>Note: this feature can result in returned data being stale by at most the specified value.</i>
AFSearchCacheInterval	The amount of time (in seconds) between refreshes of cached search results.	When the same search occurs frequently, but the results are not expected to change often, higher values give better performance.
AFSearchCacheTimeout	The amount of time (in seconds) to wait before clearing a cached search result.	When the same same search occurs frequently, higher values give better performance.
AFSearchPageSize	The page size used to retrieve results from the AF server.	If your application typically returns very few results for searches, then a small page size gives better performance. If your application typically returns many results for searches, then a large page size gives better performance.
ChannelPollingInterval	How often (in milliseconds) each Channel will notify listeners of PI/AF changes.	Reduce network traffic when changes to a monitored resource occur in rapid succession.
DisableWrites	Prevents the PI Web API from writing to PI/AF.	Increase security for publicly-accessible PI Web API systems.
PreflightMaxAge	How long (in seconds) a CORS preflight request can be cached by clients.	When CORS is enabled, increasing this value will give better performance on high-latency connections for well-behaved clients. <i>Note: if CORS settings change, this setting can reduce security until applications refresh their cached configurations.</i>
RateLimitDuration	A period of time (in seconds) that a client is bound by the RateLimitMaxRequests.	Reduce network traffic and PI System load due to poorly behaved or malicious clients.
RateLimitMaxRequests	A maximum number of requests per client (IP address) over a period of time.	Reduce network traffic and PI System load due to poorly behaved or malicious clients.
WebIDType	Changes the default Web ID type that PI Web API will respond with.	Setting this value to IDOnly will give better performance, at the cost of reduced AF hierarchy flexibility.
WebIDVersion (2018 R2)	Changes the default Web ID version that PI Web API will respond with.	While migrating systems to newer versions of the PI Web API, changing this value will prevent legacy applications from encountering unexpected types of Web IDs. <i>Note: many newer features (ex. Notifications) require Web ID 2.0. These features will stop working if this value is changed.</i>

Suspect Endpoints

- Highly suspect:
 - AnalysisRules/{webId}/AnalysisRules
 - Attributes/{webId}/Attributes
 - Elements/{webId}/Attributes
 - EventFrames/{webId}/Attributes
- Sometimes suspect:
 - Non-adhoc StreamSet calls