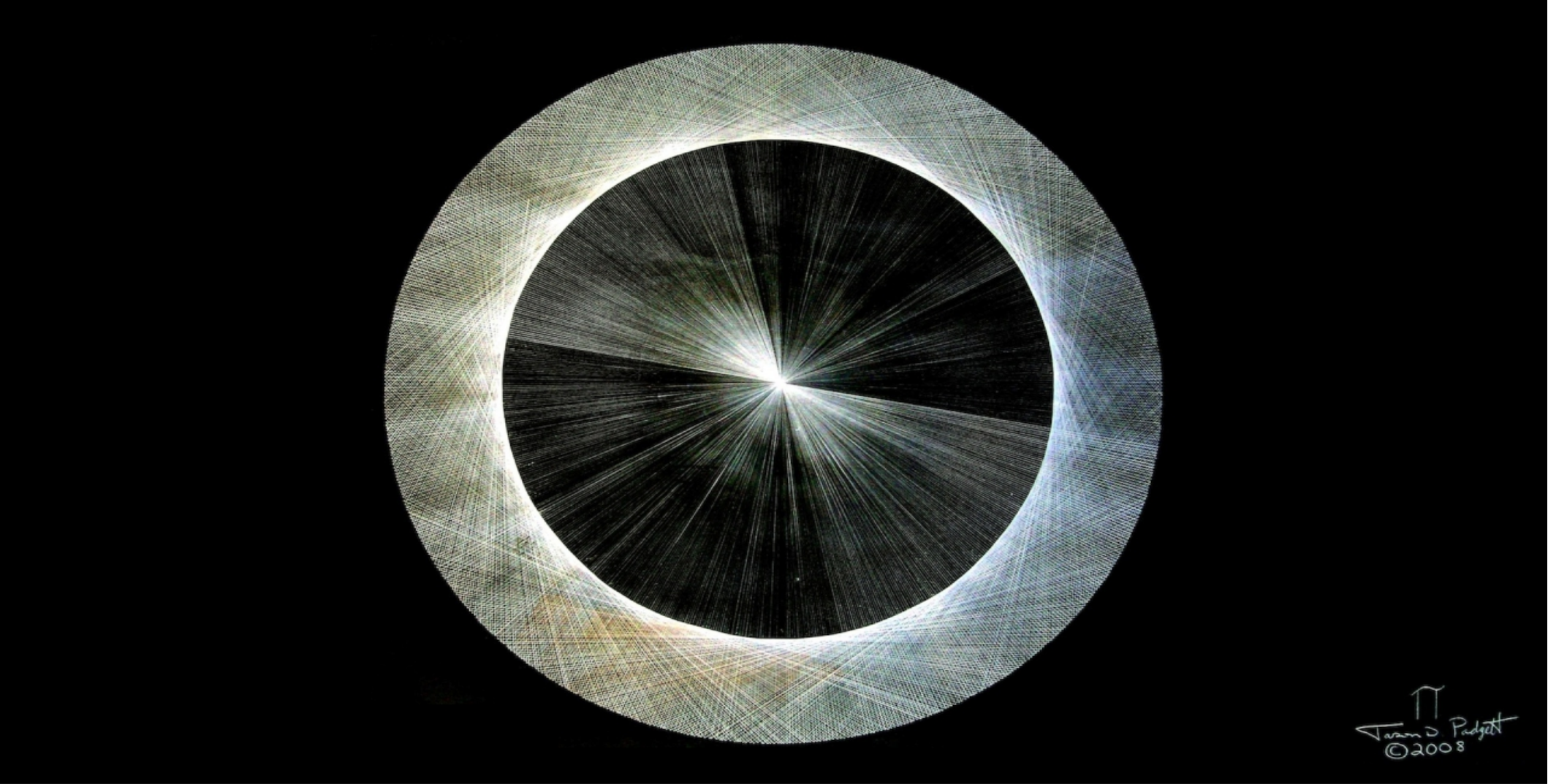

James Padgett
©2008

π v0.5

The basic advice regarding response times has been the same for thirty years [Miller 1968; Card et al. 1991]:

0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.

1.0 second is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but *the user does lose the feeling of operating directly on the data*.

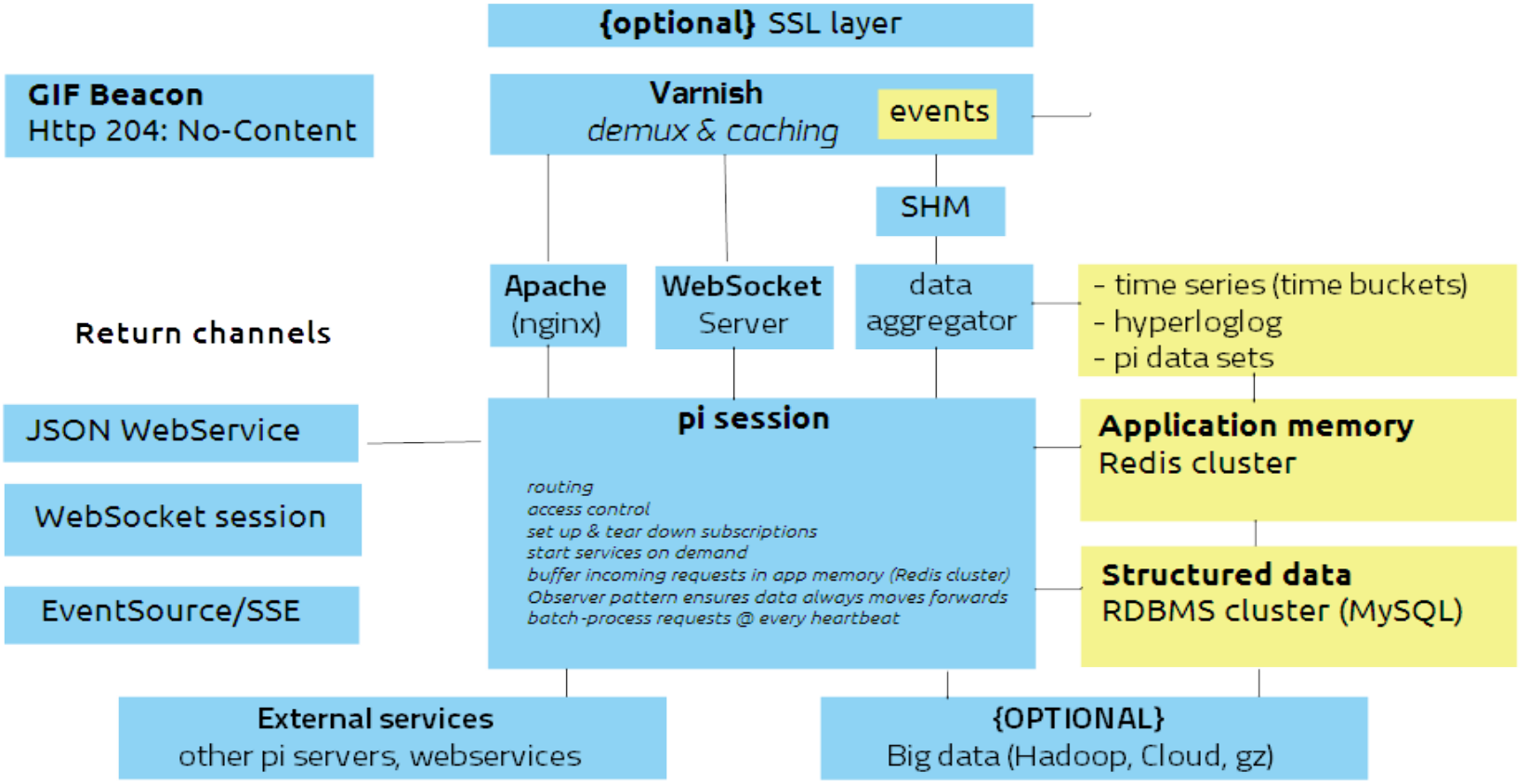
10 seconds is about the limit for keeping the user's attention focussed on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is *especially important if the response time is likely to be highly variable, since users will then not know what to expect.*

Out With the Old, In With the New

- **OUT**
- AJAX
- XML
- MVC/MVV
- data binding
- callback pasta
- Node.js
- NoSQL
- Jquery

- **IN**
- WebSockets
- JSON
- Flow-Based Programming
- Self-contained components
- WebWorkers & messaging
- Redis pubsub
- RDBMS with table templates
- HTML5 native functions
- HTML5 data types

The stack


{optional} SSL layer
GIF Beacon
Http 204: No-Content
Varnish
demux & caching
events
SHM
Apache
(nginx)
WebSocket
Server
data
aggregator
- time series (time buckets)
- hyperloglog
- pi data sets
Return channels
JSON WebService
WebSocket session
EventSource/SSE
pi session
routing
access control
set up & tear down subscriptions
start services on demand
buffer incoming requests in app memory (Redis cluster)
Observer pattern ensures data always moves forwards
batch-process requests @ every heartbeat
Application memory
Redis cluster
Structured data
RDBMS cluster (MySQL)
External services
other pi servers, webservices
{OPTIONAL}
Big data (Hadoop, Cloud, gz)

HTML5 Web Storage

- With HTML5 local storage, a larger amount of data (initially, 5MB per application, per browser) can be persistently cached client-side, which provides better performance and a better user experience .
- *Note: The 5MB maximum applies to local storage only, not to session storage, which is limited only by system memory.*

- **Benefits of local storage:**
- - Reduces network traffic
- - Significantly speeds up display times
- - Load cached data on startup (faster startup)
- - Restore application state on login
- - Allows offline use of application

LocalStorage and SessionStorage

HTML5 Web Storage defines two types of key-value storage types: sessionStorage and localStorage. The primary behavioural difference is how long the values persist and how they are shared. The following table shows the differences between the two types of storage.

LocalStorage
- 5MB per app per browser.
- According to the HTML5 spec, this limit can be increased by the user when needed; however, only a few browsers support this
- On disk until deleted by user (delete cache) or by the app
- Shared across every window and tab of one browser running same web app
- String only, as key-value pairs

SessionStorage
- Limited only by system memory
- Survives only as long as its originating window or tab
- Accessible only within the window or tab that created it
- String only, as key-value pairs

All-in on HTML5

- Works on *all touch devices*
- Works on Windows Metro, maybe
- Works on ***Adobe AIR***
- Works in any modern browser
- Works on *most Smart-TVs*

A New Idea: *π*

- **π** is a simple, lightweight, flexible and fast application platform for HTML5 compatible browsers.
- It features:
- - modular loading and dependency injection of JS and CSS
- - session support
- - utility libraries
- - a set of conventions for how to implement an interface
- - a visual component library called *pcl*
- - a *pcl* charting component library using SVG and ***raphaël.js***
- - live streaming JSON data packets over SSE (Server-Sent Events) / EventSource

Key Take-Aways

- JSON messaging connects *users*, applications, **services**, and components
- The pi address space serves as an addressing/scoping mechanism
- for events and data.
- Built-in user, authentication, session, dependency management
- Dependable, **proven technology** put together in a scaleable way
- *All-in* on **HTML5**

GIF beacon

- Encode event object in GIF url
- Request the url with an Image object, do not touch the DOM
- Catch encoded events in Varnish (with **VCL** or **C/C++**)
- Return HTTP 204 No-Content to save one HTTP packet
- Read 204 events from Varnish SHM into aggregator (using varnishlog)
- Publish live aggregates to application memory (redis)
- Batch-insert aggregated data into DB at intervals
- Collect and aggregate huge amounts of data from huge amounts of sessions

WebService API

- JSON WebServices
- With PHP session at backend
- Good for collecting sparse or bulk data from a large number of authenticated users / sessions
- Is meant mostly for transactions and batch-type services

WebSocket API

- JSON messaging
- Full duplex connection
- With PHP session
- Low latency
- No gzip
- Binary message support (in some browsers)

SSE/EventSource API

- JSON event stream
- With PHP session
- Low latency
- Gzip
- No return channel
- Good for streaming the same data to multiple clients
- Browsers manage connection

Scalability

- Incoming: Put HAProxy in front of several Varnish servers
- Internal: Use external DB and Redis clusters
- Future: ZeroMQ routing patterns

Philosophy

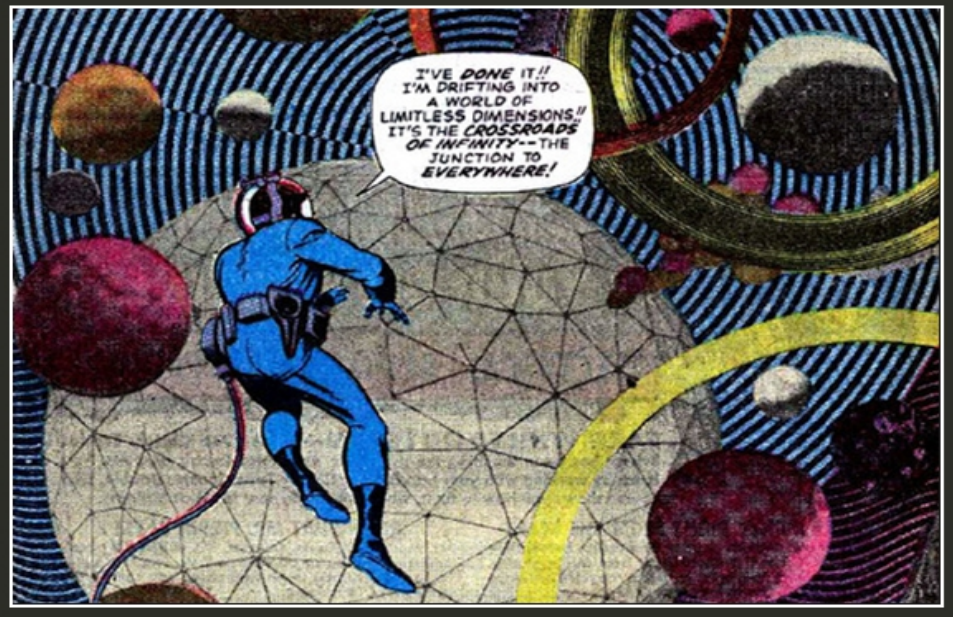
- On-demand and Just-in-time (**JIT**)
- Batch processing & background processing
- **The Observer Pattern** *«Keep moving forward»*
- Self-contained components/subsystems
- Follow conventions, use best practices

The Future

- Go binary (JS typed arrays)
- Go compiled or LLVM (C / C++ / Python / Go / ObjectPascal)

The End Station

- Go any-to-any binary with
- ZeroMQ


I'VE DONE IT!! I'M DRIFTING INTO A WORLD OF LIMITLESS DIMENSIONS!! IT'S THE CROSSROADS OF INFINITY--THE JUNCTION TO EVERYWHERE!