Community Forum

Fast Dash Apps cheat sheet

Browser: Rendering Components	>10K SVG/HTML *rendered* elements	"Less is More" - Render less components - Multi-Page Apps - Render on-the-fly (pagination, filtering) - Aggregate data - Sample Data - Summary Statistics SVG -> WebGL -> Datashader Serverside Store Move Dash closer to Users Clientside Callbacks Compression (gzip)
	>1M WebGL points	
Network: Browser Client <> Python Server	1-50MB Input/Output/State	
	Slow Network	
Dash Python Server	Insufficient Workers & Request Handling	More workers or replicas! gunicorn app:server --workers 8 --preload Background Callbacks Faster Requests -> More throughput Multi-Output Callbacks & Prevent Initial Callback Clientside Callbacks
	Slow Number Crunching	Faster Algorithms Cache (memoize) Data Aggregation: Pandas -> Vaex -> SQL Computation: Lists -> Numpy -> Numba Parallelize: Multi-processing/GPU/Vaex/Ray/Dask/Spark Faster Hardware In-Memory vs Disk Various Python Tricks: (List Comprehensions, O(1) Data Structures, Python 3.11 (October!))
	Waiting for Data	Aggregate & cache in memory when app boots Aggregate & cache on background schedule
Network / Data Transfer: Dash Server <> Data Store	>200MB-1GB Data Transfer	Perform computations in database instead of in Python memory Aggregate & cache in advance when app boots Aggregate & cache on background schedule
	Slow Data Transfer	Cache closer to Dash app (filesystem, Redis)
		Move Dash closer to Data Store Compression for Network Faster Deserialization (HDF5, Arrow)
Data Store	Insufficient # Connections (Queued Queries)	Connection Pooling Aggregate & cache
	Query is Slow	Improve Query: Index, Query Optimization, Materialized Views Faster Data Store (Row -> Column) Faster Disk