Cheat Sheet - Performance Principles, Patterns and Anti-Patterns

Principle	Description
Performance Objective Principle (POP)	Define specific, quantitative, measurable performance objectives for performance scenarios.
Instrumenting Principle (IP)	Instrument systems as you build them to enable measurement and analysis of workload scenarios, resource requirements, and performance objective compliance.
Centering Principle (CP)	Identify the dominant workload functions and minimize their processing.
Fixing-Point Principle (FP)	For responsiveness, fixing should establish connections at the earliest feasible point in time, such that retaining the connection is cost effective.
Locality Principle (LP)	Create actions, functions, and results that are close to physical computer resources.
Processing vs. Frequency Principle (PFP)	Minimize the product of processing times frequency.
Shared Resources Principle (SRP)	Share resources when possible. When exclusive access is required, minimize the sum of the holding time plus the scheduling time.
Parallel Processing Principle (PPP)	Execute processing in parallel (only) when the processing speedup offsets the communication overhead and resource contention delays.
Spread-the-Load Principle (STLP)	Spread the load when possible by processing conflicting loads at different times or in different places.

Pattern	Description	Principle(s)
Fast Path	Identify dominant workload functions and streamline the processing to do only what is necessary	CP
First Things First	Focus on the relative importance of processing tasks to ensure that the least important tasks will be the ones omitted if everything cannot be completed within the time available	FP
Coupling	Match the interface to objects with their most frequent uses	CP, LP, PFP
Batching	Combine requests into batches so the overhead processing is executed once for the entire batch instead of for each individual item	PFP
Alternate Routes	Spread the demand for high-usage objects spatially, that is, to different objects or locations	STLP
Flex Time	Spread the demand for high-usage objects temporally, that is, to different periods of time	STLP
Slender Cyclic Functions	Minimize the amount of work that must execute at regular intervals	CP

Antipattern	Problem	Solution	Principle Violated
"god" Class	Occurs when a single class either 1) performs all of the work of an application or 2) holds all of the application's data. Either manifestation results in excessive message traffic that can degrade performance.	Refactor the design to distribute intelligence uniformly over the application's top-level classes, and to keep related data and behavior together.	LP, PFP
Excessive Dynamic Allocation	Occurs when an application unnecessarily creates and destroys large # of objects during its execution. Overhead in creation and destruction of objects degrades performance	1) Recycle objects (object pool) rather than creating new ones each time. 2) Use Flyweight pattern to eliminate need to create new objects	FP, LP, PFP
Circuitous Treasure Hunt	Occurs when an object must look in several places to find the information that it needs. If a large amount of processing is required for each "look" performance will "suffer"	Refactor design to provide alternative access paths that do not require a Circuitous Treasure Hunt (or to reduce the cost of each "look")	LP, FP
One-Lane Bridge	Occurs at a point in execution where only one, or a few, processes may continue to execute concurrently (e.g., when accessing a database). Other processes are delayed while they wait for their turn.	To alleviate the congestion, use the Shared Resources Principle to minimize conflicts	SRP, STLP
Traffic Jam	Occurs when one problem causes a backlog of jobs that produces wide variability in response time which persists long after the problem has disappeared	Begin by eliminating the original cause of the backlog. If this is not possible, provide sufficient processing power to handle the worst-case load.	SRP, STLP

Based on Addison Wesley's:
Performance Solutions - A Practical Guide to Creating Responsive, Scalable Software
Connie U. Smith and Lloyd G. Williams

Cheatsheet created by: Mohit Soni (http:// mohitsoni.com)