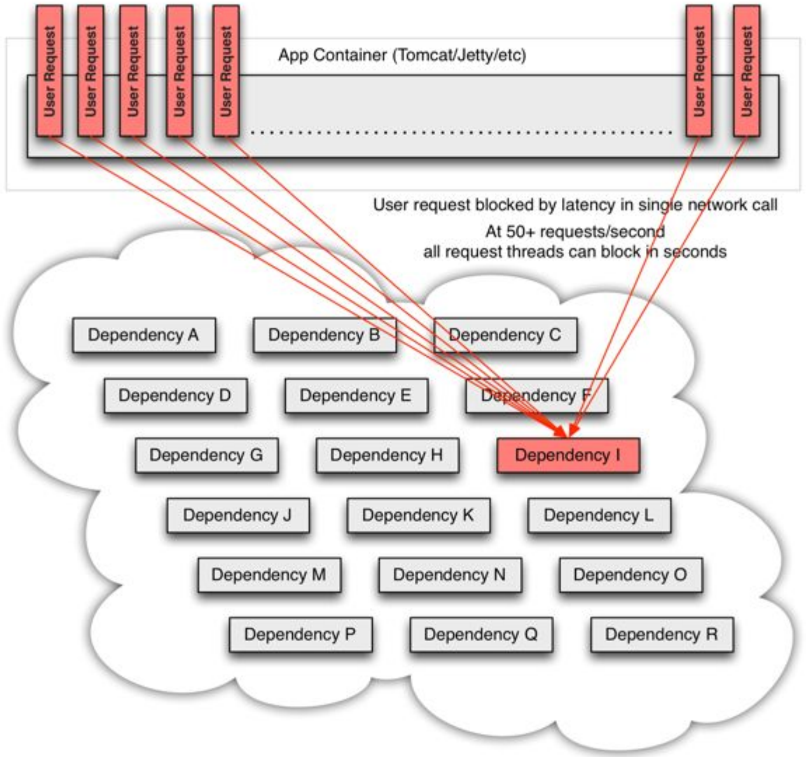
Pivotal

Bulkhead Pattern

Problem: Failure Cascades to Caller

- *Slow Responses* may contribute to *Blocked Threads.*
- Calling application's thread pool is exhausted waiting on misbehaving dependency. This is a *Chain Reaction* induced by *Blocked Threads.*
- Failure cascades to caller, resulting in *Cascading Failure*
- It may be a good idea to Isolate parts of our system to mitigate failures.



Solution - Bulkheads

In preceding examples, if we could not have used Timeouts:

- Client thread pools could become depleted
- Could cause *Cascading Failure* on app containers
- We want to protect client Thread Pools
- We can use *Bulkheads* through dedicated Thread Pools

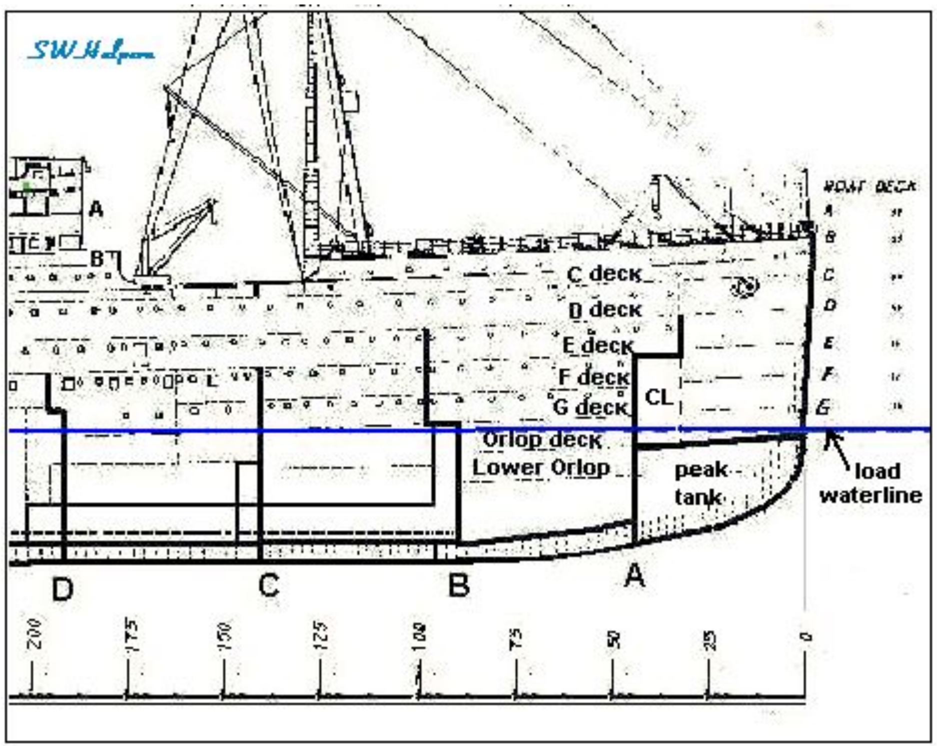
Bulkheads

What?

- Mechanism to isolate parts of a system

Why?

- Isolate points of failure
- Limit scope of failure



Pivotal

Bulkhead examples

Hardware, platforms:

- Resource Pools
  - Geographic: Datacenters
  - Hardware: Racks, Enclosures, Servers, CPU, Core, Hardware Threads
  - Virtualization: Hypervisors, Containers
- Network Partitions

Software:

- Thread Pools
- Processes

Solution - Bulkheading through Thread Pools

In preceding examples, if we could not have used Timeouts:

- An application’s thread pools could become depleted
- Could cause *Cascading Failure* on app containers
- We want to protect application’s thread pools
- We can use *Bulkheads* through separate dedicated thread pools from the default application thread pools
- When exhausting dedicated thread pools, reject call on the protected command

Solution - Bulkheading through Semaphores

You may not need full isolation through thread pools:

- Thread pools add resource overhead
- Use of thread pools may not be possible with call client thread state.
- Track concurrent request count through semaphore.
- When reaching maximum allowed count, reject call on the protected command

Load Shedding and Tradeoffs

- The act of rejecting requests is known as “Load Shedding”
- Consequence of using Thread or Semaphore isolation bulkheads
    - You must handle the rejection through a fallback