Condos and Clouds

Thinking about Cloud Computing
by Looking at Condominiums

Pat Helland

HPTS Oct 2011

Outline

- Introduction
- The Drive towards Commonality in Buildings
- A Useful Pattern for SaaS Applications
- Multi-Tenancy: Rules to Make It Work
- Conclusion

I Live in a Condo…

Our home (until a few weeks ago)!
Great view…. But my ears pop going home!

- Not everyone wants to live in a condo
  - Not good for kids playing in the backyard, dogs running in the backyard, working on cabinetry in your garage, or having a garden
- But it has its advantages!
  - Common heating/air-conditioning "just works"
  - Infinite supply of hot water for long showers
  - Someone else takes the trash out "to the street"
  - It comes with a building engineer who fixes most things!

Sometimes, It's Really Nice to Outsource a Bunch of Hassles

In Exchange, You Live with Some Constraints

Constraints and Concierge Services

Constraints and Concierge Services in Buildings

Building Type	Services	Constraints
Housing	• Reservations • Package/Dry-Cleaning • Shared Exercise & Pool • Shared Engineering	• Limits on Parking, Noise, Pets, & BBQing • No Garage Projects • No Gardening Projects
Office	• Shared Bathrooms • Shared Lobby • Shared Copiers/Coffee (?) • Shared Engineering	• Fixed Office Layout • No Sleeping at Work • (Typically) No Pets • (Maybe) Dress Code
Retail Mall	• Shared Engineering • Shared Parking • Shared Common Space • Shared Security	• Common Mall Hours • Only Retail Activities • Probably Constraints on the Type of Retail

What Are the Constraints and Concierge Services in Cloud Computing?

What Can the Shared Infrastructure Do to Make Life Better for a Sharing App?

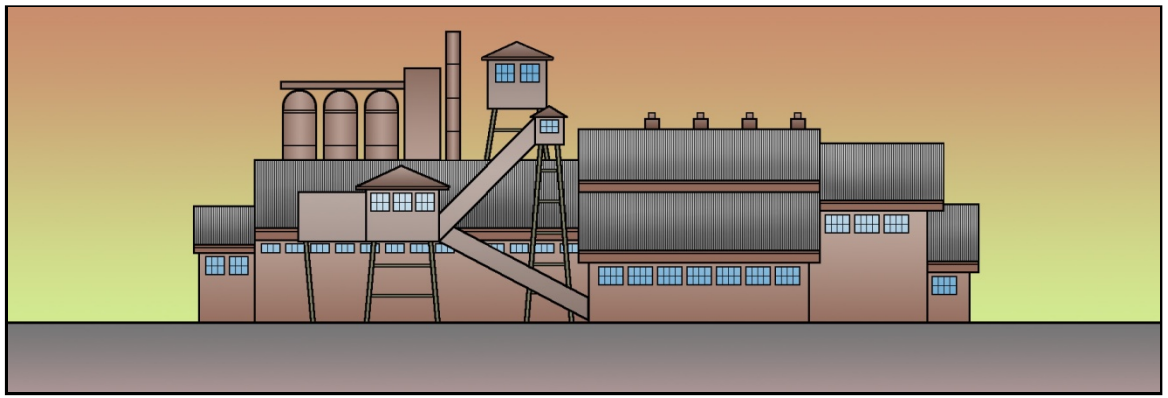
What Constraints Must a Sharing App Live within to Fit into the Shared Cloud?

Outline

- Introduction
- The Drive towards Commonality in Buildings
- A Useful Pattern for SaaS Applications
- Multi-Tenancy: Rules to Make It Work
- Conclusion

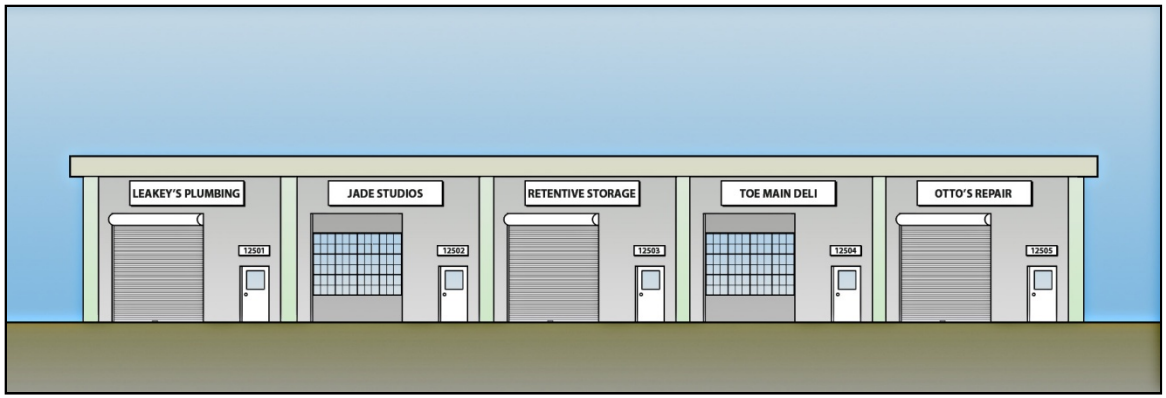
Specialized Buildings

- Heavy manufacturing requires special buildings
  - Steel mills, shipyards, auto factories, airplane manufacturers
  - Custom-made structures are the only way to make these products
- Some retail requires special buildings
  - Costco, Wal-Mart Super-stores, Home Depot, IKEA
- *Each has such unique requirements that no standard building can suffice*



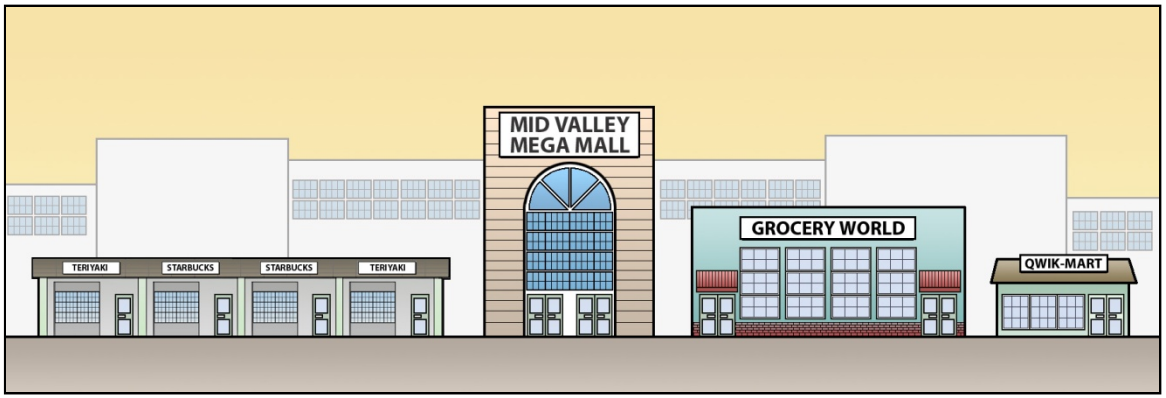
Convergence In Industrial Parks

- Industrial parks have standard space
  - Big roll up doors
  - Simple space: concrete floors, basic walls, high roofs
    - One small bathroom stall
- Many different kinds of business live here
  - Light manufacturing
  - Light distribution
  - Automotive repair
  - Specialized retail
  - Lunch shops
  - Low end office space



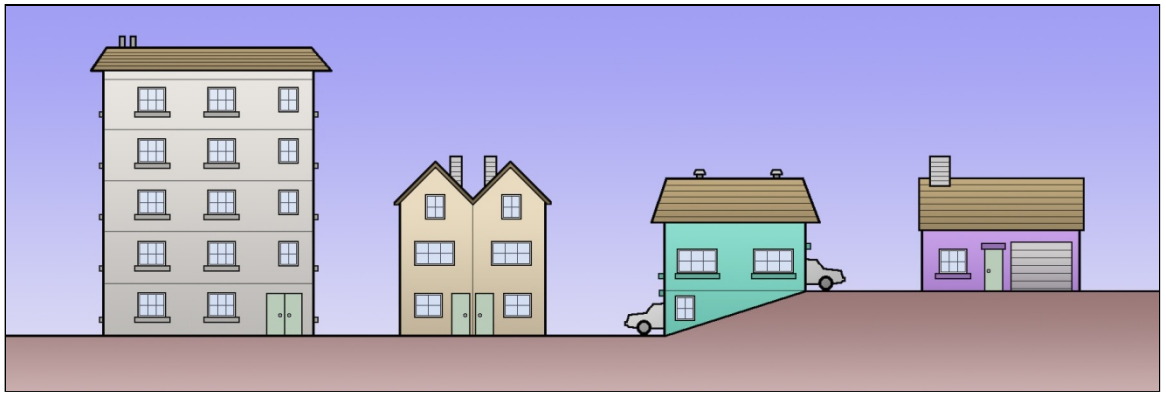
Convergence in Retail Buildings

- Different forms of retail
  - Retail strip malls
    - Standardized and inexpensive retail space
    - Outsource site, structure, skin, infrastructure
  - Mega-malls
    - Standardized and expensive retail space
    - Outsource site, structure, skin, infrastructure, mall-branding
  - Grocery stores
    - Specialized retail space (unique form for grocery stores)
    - May be reused across grocery store chains
- Reuse of standardized retail space
  - Many different companies can use the space
  - Leasehold improvements within the space



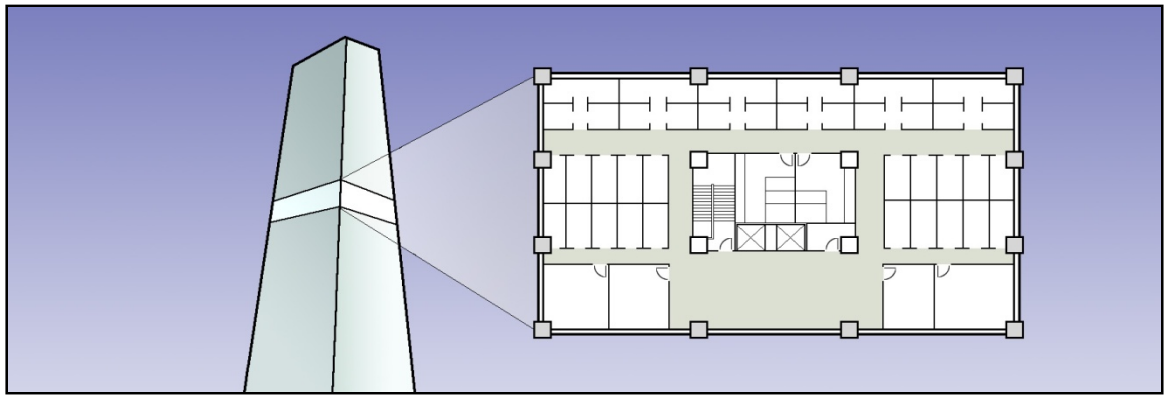
Convergence in Housing

- Housing is about people
  - Many forms of housing
  - Contains people and their stuff
- Convergence in housing
  - Standardized door sizes, plumbing, electrical connections, etc.
    - Your stuff works in the new home
  - Just move your stuff in and unpack
- Many differences in housing forms
  - Designed for compatibility with people and their stuff



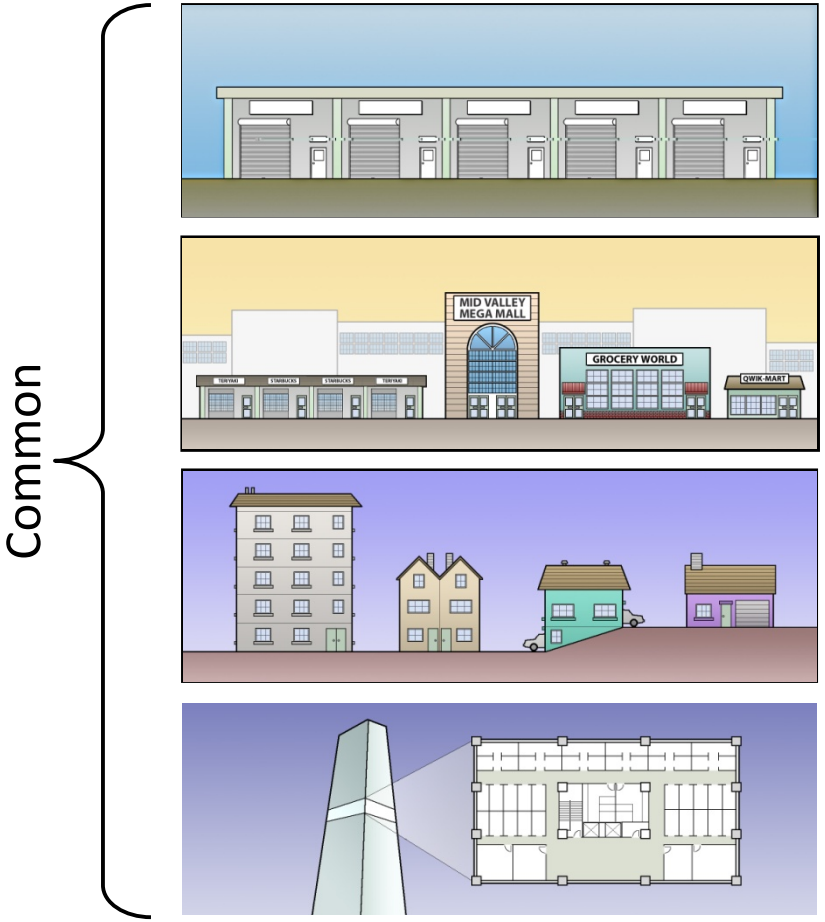
Convergence in Office Space

- Office work is about people and information
  - More about people moving in and out
  - Less about stuff moving in and out
- The building requirements for office needs are consistent
  - Standard need for coffee machines, rest rooms, and copiers
- Standardization of office space is important
  - Both skyscrapers and suburban ("edge-city") sprawl
  - Outsource site, structure, skin, and building infrastructure
  - Customize space plan and stuff



The Drive Towards Commonality in Buildings

- ***Buildings Are Created Without Knowing Who Will Use Them!!***
  - Industrial parks, retail, office space, and housing
  - Each has standard specifications for occupants
  - New occupants fit into the space of the building…



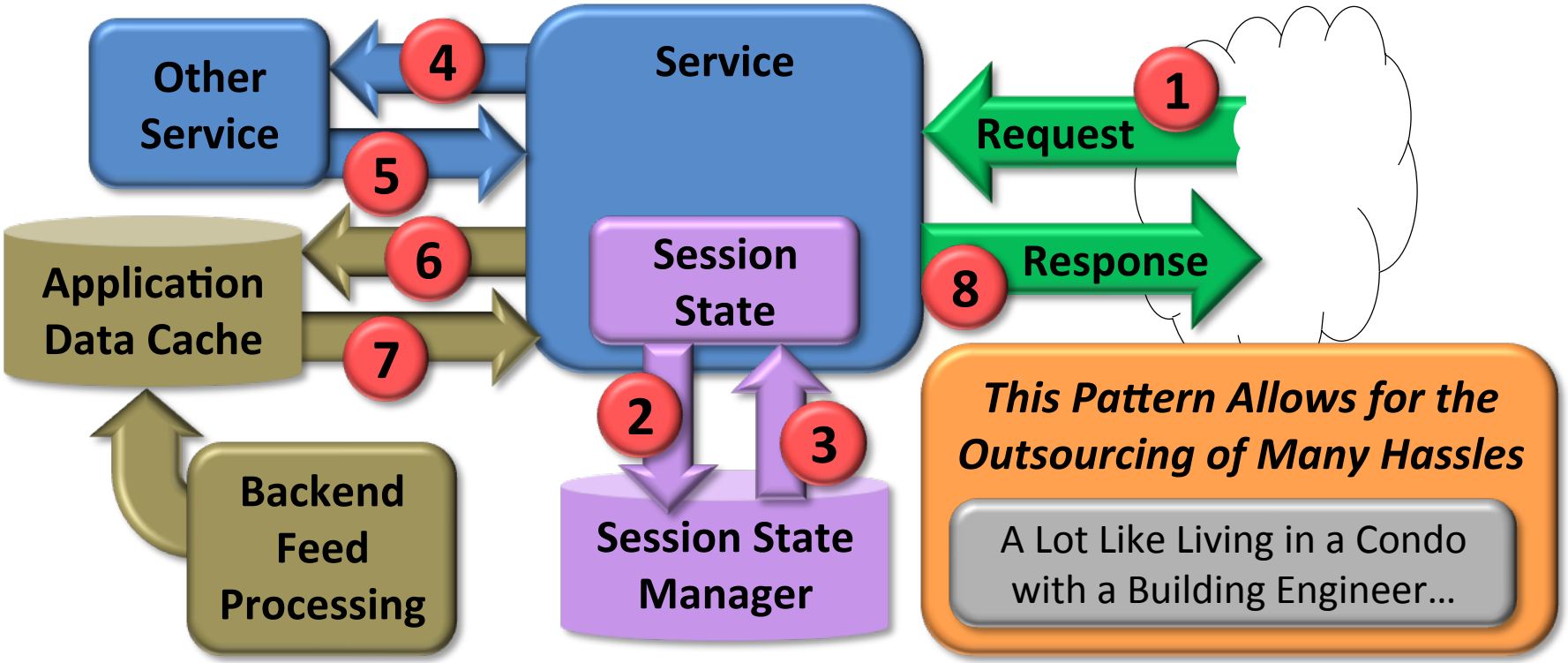
- Custom buildings occur
  - Less and less common
  - Sometimes they are needed!

Outline

- Introduction
- The Drive towards Commonality in Buildings
- A Useful Pattern for SaaS Applications
- Multi-Tenancy: Rules to Make It Work
- Conclusion

Many Service Apps Fit a Pattern

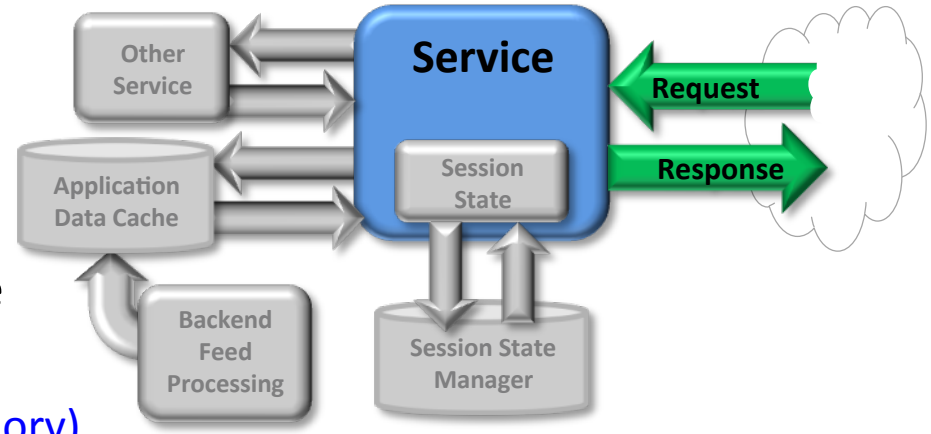
- Many Service applications follow this pattern:
  - Requests come in from the web to a service
  - You optionally fetch state associated with this session
  - The app may or may not invoke other services
  - The app may access a data cache populated from feeds/back-end processing
  - A response is sent back to the user



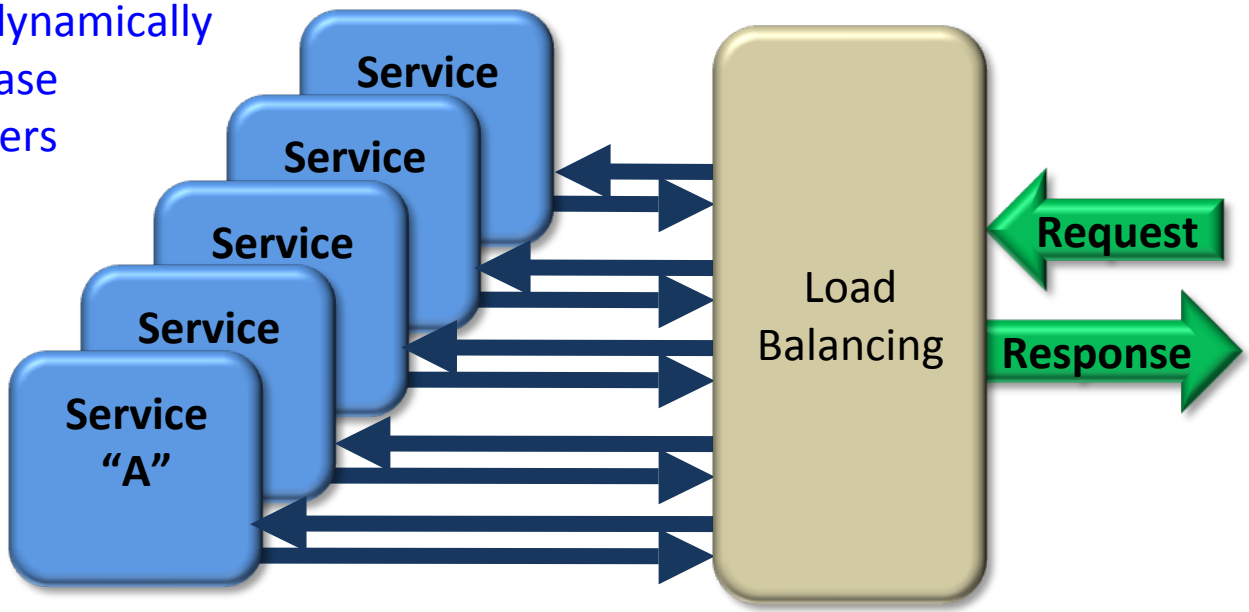
The Auto-Magic Dream (Concierge Services)

Auto-Scaling	As the workload rises, additional servers are automatically allocated for this service. Resources taken back when load drops.
Auto-Placement	Deployment, migration, fault boundaries, and geographic transparency are all included. Applications are blissfully ignorant.
Capacity Planning	Analysis of traffic patterns of service usage back to incoming user work load. Trends in incoming user workload are tracked.
Resource Marketplace	Plumbing tracks a service's cost as it directly consumes resources and indirectly consumes them (by calling other services).
A/B-Testing and Experimentation	Plumbing makes it easy to deploy a service on a subset of the traffic and compare the results with the previous version.
Auto-Caching / Data Distribution	Data is fed into a datastore and processed there. This processed data is cached for easy access by services.
Session State Management	User session information is captured before a service completes. The next request easily fetches the state to use.

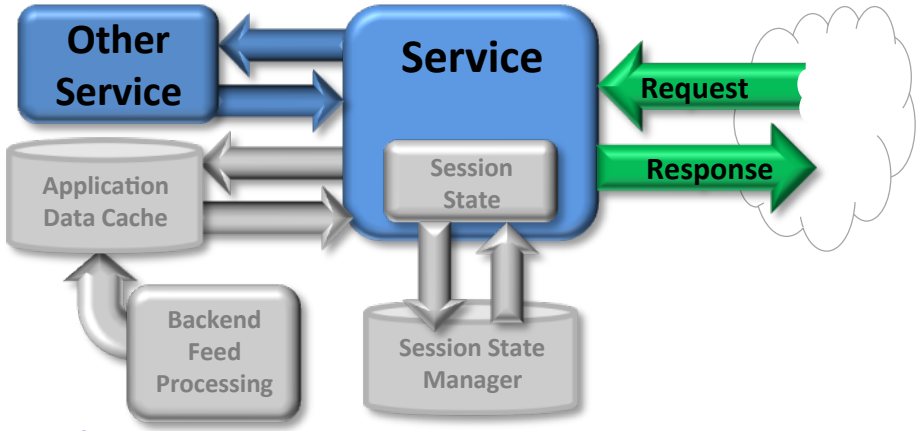
Stateless Request Processing



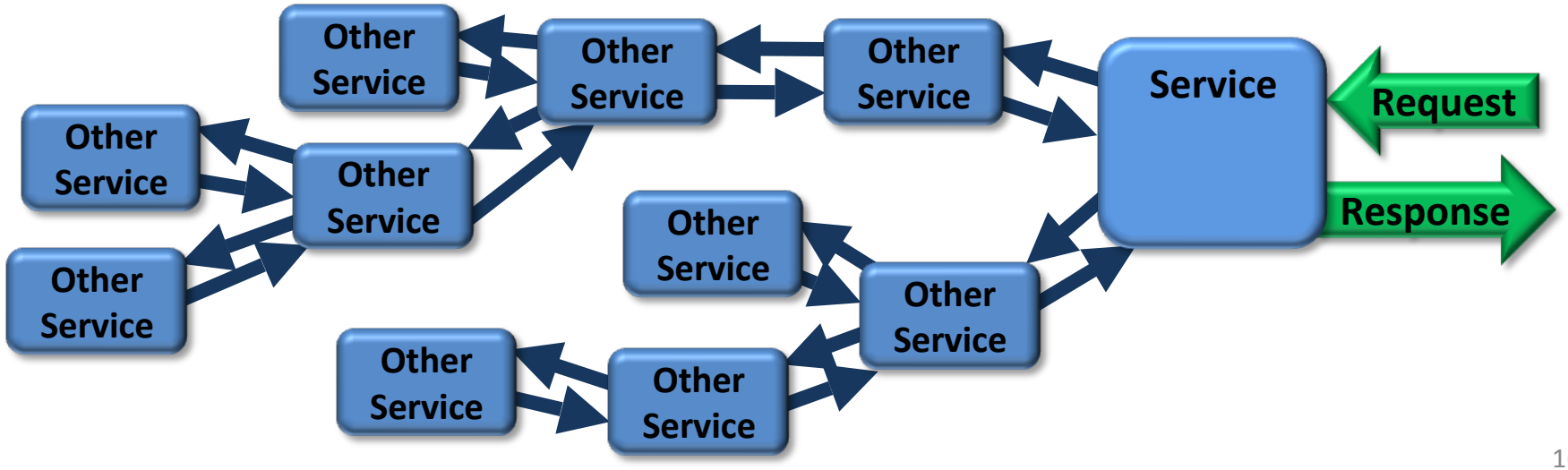
- Incoming requests are routed to one of the copies of the service
  - On arrival, there is no state (or memory) associated with the session
  - *At this point*, we consider the service to be stateless (it may get state later)
- The plumbing keeps a pool of servers capable of implementing the service
  - Incoming requests are dynamically routed to these services
  - The plumbing will dynamically increase and decrease the number of servers implementing this service as needed



Composite Request Processing



- Frequently, a service calls other services to get its job done
  - These may or may not grab session state from the session state manager
- The composite call graph may get complex
  - Sometimes there will be cycles
    - The service must know when to break the cycle
  - Sometimes the depth of the call graph may get very deep
- All of these service calls will need to complete to build the user response
  - The work fans out, processes, and fans back



SLAs and Request Depth

SLA: Service Level Agreement

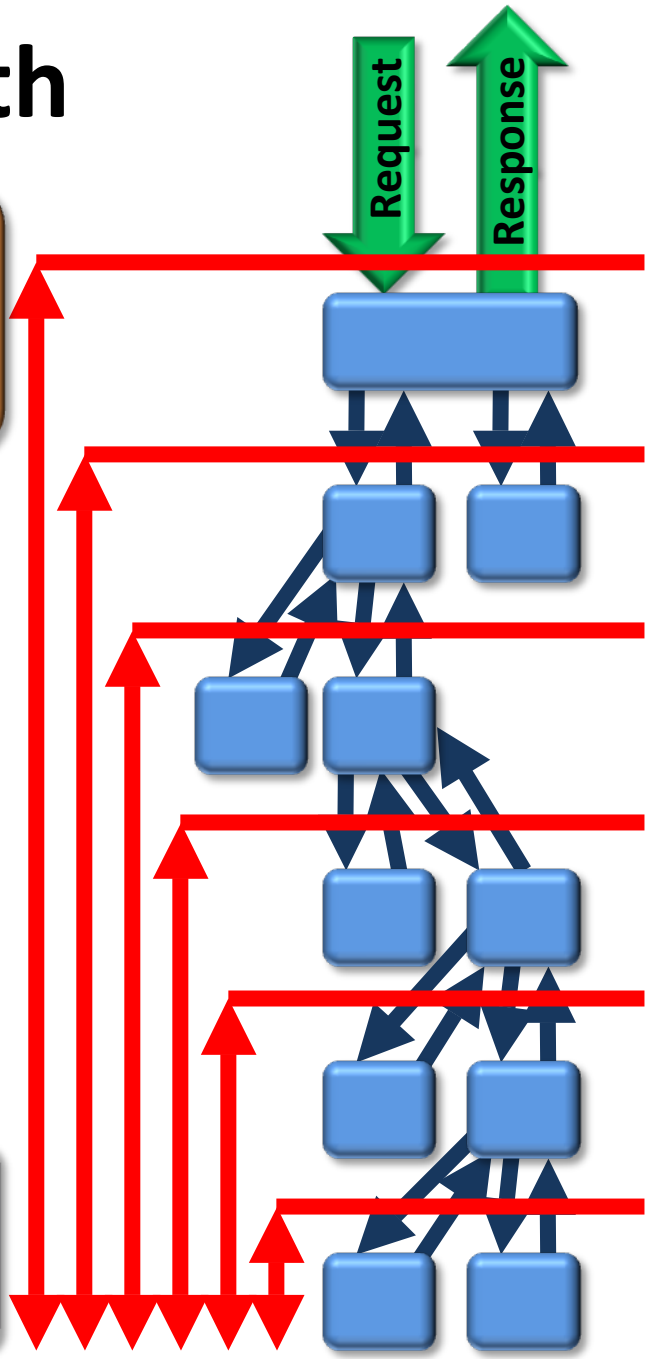
Does the "Service" Provide the Level of Service It Promised?

Is the Response Fast Enough?

- System-wide SLAs need tight component SLAs
  - Multi-level call graph in service structure
- Example SLA:
  - 300ms response for 99.9% of requests with 500 requests per sec
- Cross-service dependencies
  - Force tight SLA responses
- Average versus Percentile
  - Average not strict enough
  - User experience unacceptable

Lots of Pressure on Services at the Bottom of the Stack!

You Need to Answer Fast and Predictably!



Pounding on the Services at the Bottom

- The Deeper the Call Stack, the Tighter the SLA
  - It's Time Is Factored in the Caller's SLA
- Consider Session-State Manager and Auto-Cache Manager
  - They get called a lot
  - They get called from services already deep in the stack!

Deeper the Call Stack → More Pressure

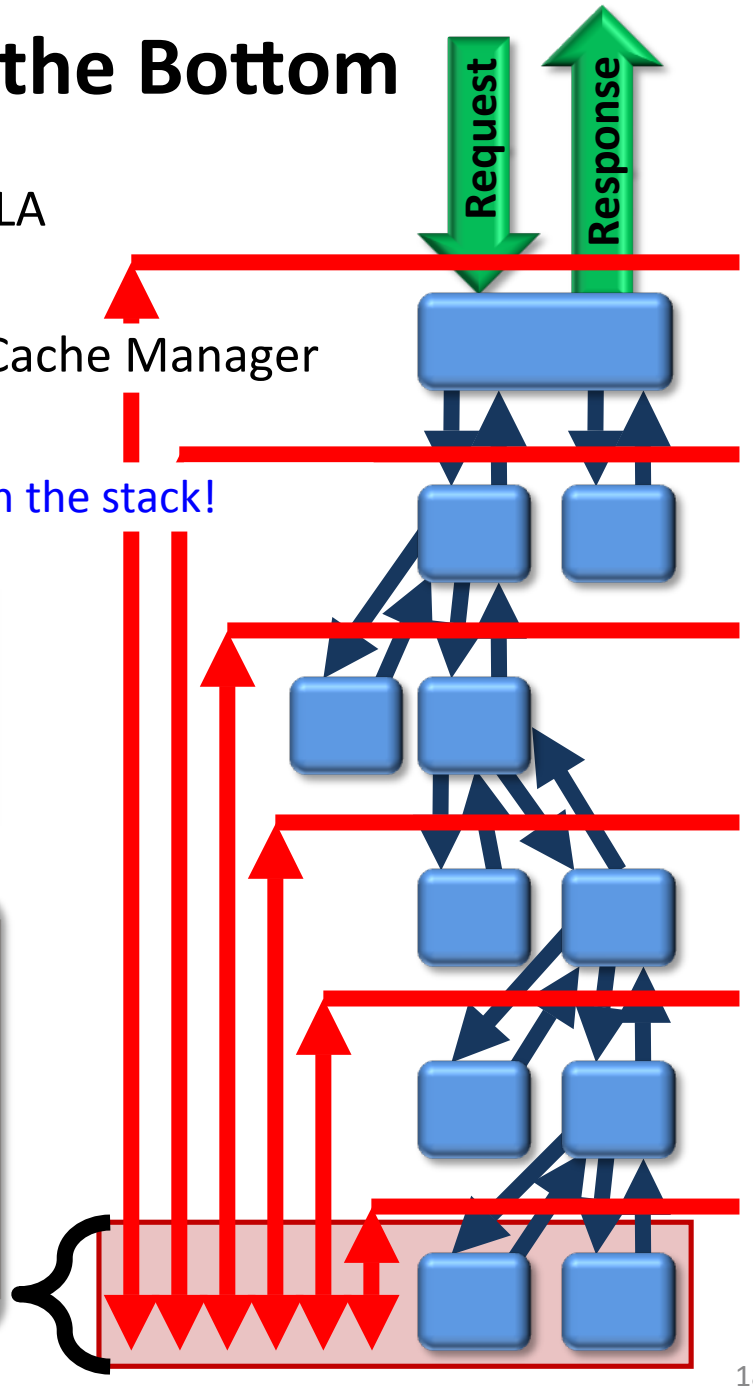
Can Result in Needing to Sacrifice Utilization!

How Are You Going to Provide a Tight SLA?

Short Minimum Response Time?

Really Low Utilization?

Both?...



Automatic Provisioning to Meet SLAs

- The plumbing can know the desired SLA for each service it manages
  - End-user facing services can have their SLAs configured to the plumbing
  - Plumbing can know which services call which other services
    - This can be determined dynamically by watching the interactions
  - Based on the demands of the calling services, a desired SLA can be calculated

It Is Unlikely that SLA Goals Can Be Fully Automated

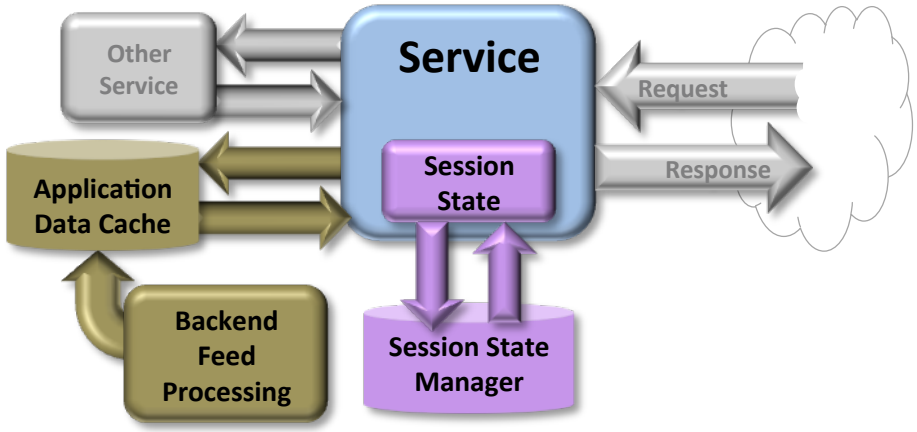
When a Calling Service Calls Many Different Dependent Services,
What Is the Relationship across Their SLAs?

- Within limits, the SLA for a service can be met by increasing the number of servers and, hence, decreasing the utilization of the server pool
  - The plumbing can dynamically increase the server pool to meet the SLA

This Technique Is Useful Sometimes but Not for Everything

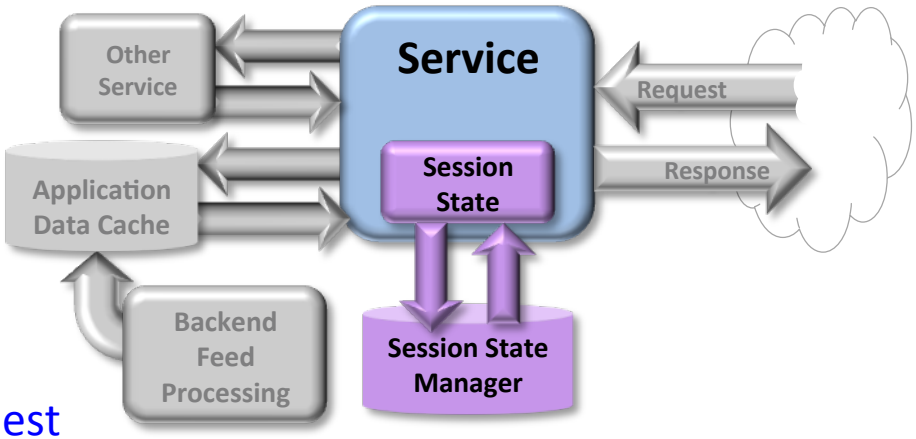
The Minimum Response Time for the Service Must Be Acceptably
Close to the Desired SLA for the Service

Accessing Data and State



- When a request lands into a service, it initially has no state other than what arrives with the request
  - It can fetch session state
  - It can fetch cached data for some application specific data item based on key
- *Session state*: fetched from the session state manager using a session key
  - When changes are made, they are stored back to the session state manager
- *Cached data*: fetched from the cache manager with domain name and key
  - Used to store data which is derived from feeds
  - This application (service) specific data is *read-only by the service*

Managing Scalable and Reliable State

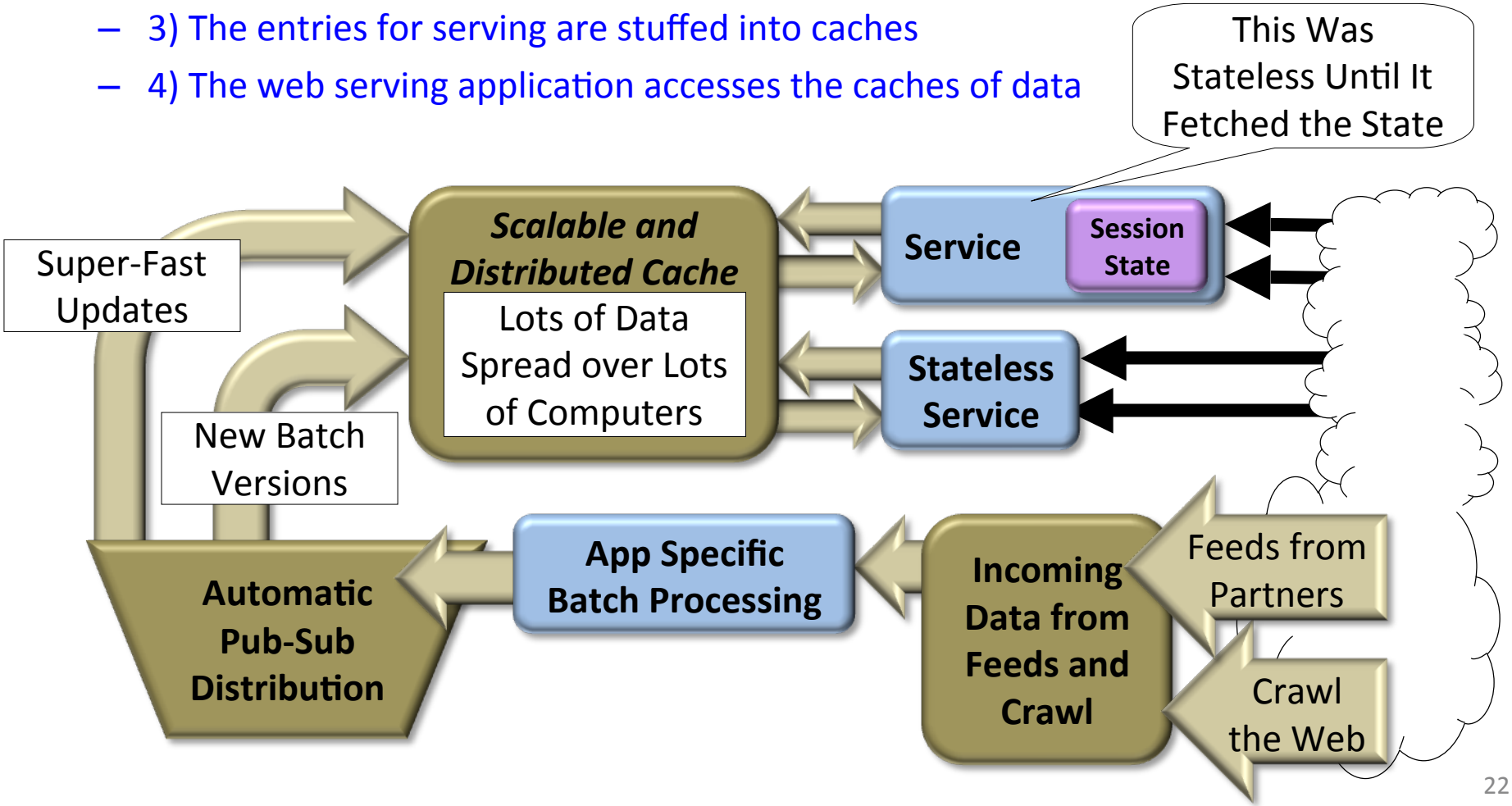


- The *Session State* is keyed by the session-id
  - The session-id comes in on the request
  - When the service calls the session-state-manager, a blob of state is returned
- The plumbing manages scaling the session state
  - The session state manager will expand as necessary when the number of sessions grows
- The plumbing manages the durability of the session state
  - It is essential that the state usually survives system failures
  - We should consider schemes which *rarely* lose updates and gain performance
- The plumbing must give excellent responsiveness

Typical Requirement: 5 ms Response 99.9% of the Time
in a First-Class Implementation of the Session State Manager

Data Publication and Updates

- The general model for data is:
  - 1) Backend processing receives data from the web by feeds or by crawling
  - 2) Application code on the backend munches the data making entries to serve
  - 3) The entries for serving are stuffed into caches
  - 4) The web serving application accesses the caches of data

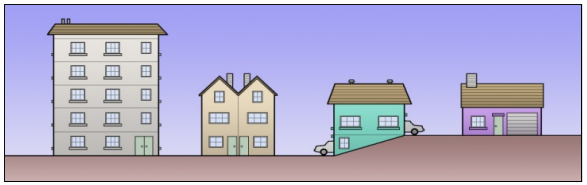


Outline

- Introduction
- The Drive towards Commonality in Buildings
- A Useful Pattern for SaaS Applications
- Multi-Tenancy: Rules to Make It Work
- Conclusion

Landlords, Management Staff, Keys, and Privacy

- In rented apartments, owned condos, or rented houses, the landlord or engineering staff typically have a key to your home
  - There are well established reasons they may enter
  - They can (possibly) get into a world of shit if they come in for other reasons…



- Renting or leasing a property (home, office, retail, or light manufacturing) implies a bidirectional trust relationship
  - Tenants will follow the rules, pay the rent, not trash the property, and not bug the neighbors
  - Landlords will grant access to the property, keep the services running, and respect the privacy of the tenants
- Landlord/Tenant laws and rules took years to establish
  - They are even more complex in condominiums with shared ownership
  - There are different laws and expectations for different usage patterns (e.g. housing, retail, office, …)

Common Carriers: Protecting the Provider and Consumer

- **Common Carrier Laws in the United States**
  - Applied first to the railroads and later to telephone companies
  - **Pricing:** A common carrier may not offer differentiated pricing across customers
  - **Protection for Customers:** A common carrier has rules for privacy, safety, care for their customer
  - **Protection for Provider:** A common carrier provides a service and is not responsible if that service is used for illegal purposes

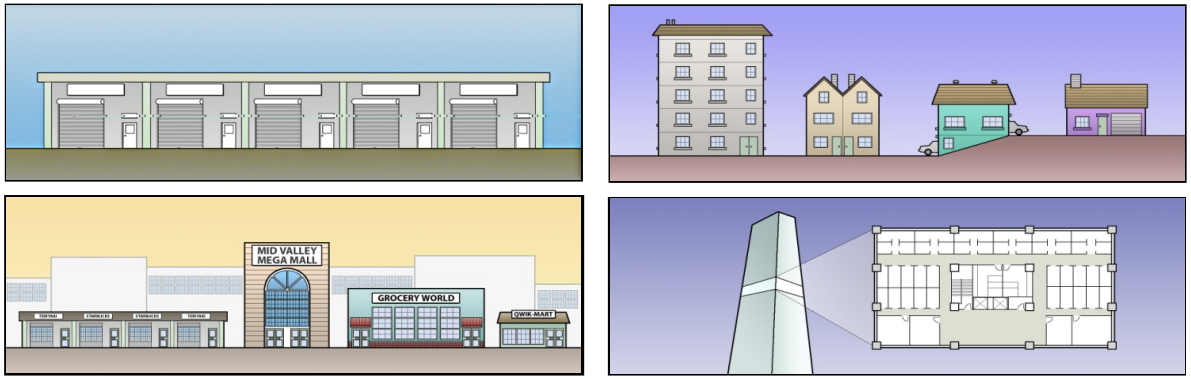
Legal Precedent Established that the Phone Company Was Not Liable as a Co-Conspirator If Two Gangsters Plotted a Bank Robbery over the Phone

- **Condominium Laws in the United States**
  - Define the rights, responsibilities, and protections for the developers of buildings
  - Define the rights, responsibilities, and protections for the owners of units
  - Define the rights, responsibilities, and protections for the homeowners' board

We Need Clearer Laws about the Rights, Responsibilities, and Protections for All Parties Involved in Cloud Computing!

Defining Constraints and Empowerment

- Programming platforms define Application Programming Interfaces (APIs)
  - These define their respective models for use and platform support
  - Different platforms may have different models for use and support
- Shared buildings have expectations for usage
  - They are (typically) built without knowing *who* will occupy them
  - They are built with an expectation of *how they will be used*
    - You are not allowed to live and sleep in a retail store or industrial park

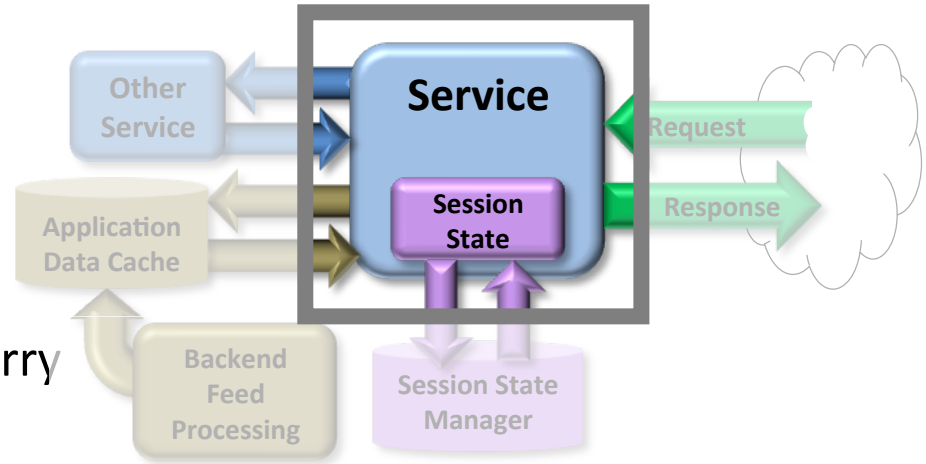


- We need to define the expectations for usage for Platform as a Service
  - It is OK for this to address a large class of customers but be useless for others
  - We are designing the platform without knowing who will occupy it
  - It must offer great "concierge services" with acceptable constraints

Outline

- Introduction
- The Drive towards Commonality in Buildings
- A Useful Pattern for SaaS Applications
- Multi-Tenancy: Rules to Make It Work
- Conclusion

Automatic Services, State, and Data



- Applications using the plumbing worry about their business code
  - They have session state, application data cache, and calls to other services
  - They simply implement their own business logic, not system issues
- These applications build their apps in two pieces:
  - Back-end feed/crawl processing:
    - Data from feeds and crawls is batch processed to make cache entries
  - Front-end web-serving:
    - Services awaken, (optionally) fetch session state, access cache, call other services, calculate their business logic, and return responses

The Plumbing Prescribes HOW to Access These Services

Session State, Cache, and Service Calls Are Done with Special Interfaces

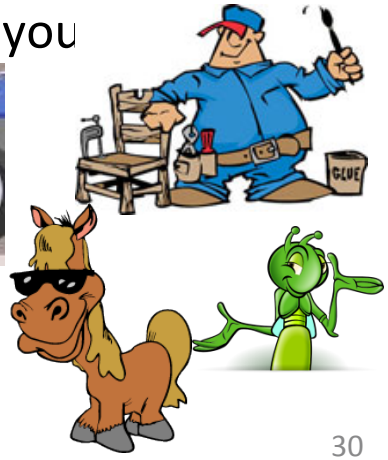
Constrained Application Functionality So the Plumbing Can Provide the Support

The Auto-Magic Dream (Concierge Service)

Auto-Scaling	As the workload rises, additional servers are automatically allocated for this service. Resources taken back when load drops.
Auto-Placement	Deployment, migration, fault boundaries, and geographic transparency are all included. Applications are blissfully ignorant.
Capacity Planning	Analysis of traffic patterns of service usage back to incoming user work load. Trends in incoming user workload are tracked.
Resource Marketplace	Plumbing tracks a service's cost as it directly consumes resources and indirectly consumes them (by calling other services).
A/B-Testing and Experimentation	Plumbing makes it easy to deploy a service on a subset of the traffic and compare the results with the previous version.
Auto-Caching / Data Distribution	Data is fed into a datastore and processed there. This processed data is cached for easy access by services.
Session State Management	User session information is captured before a service completes. The next request easily fetches the state to use.

Condos Impose Constraints

- My brother raises pet chickens for eggs… He doesn't live in a condo
  - My condo doesn't allow pet chickens.
- Condo buildings are designed with an expectation of their usage pattern
  - People living with a certain density, certain lifestyle, and certain restrictions
  - Many services are preplanned:
    - We have an exercise room, concierge, wine storage, parking, and a lovely set of rooftop patios for barbequing with nice gas grills and gas fireplaces
- It is the constraints that allow the community to exist
  - Shared services, shared building, shared maintenance, and shared expenses
  - Condos are based on sharing for reduced costs and increased benefits
- If the constraints aren't right for you, condo living isn't right for you
  - If you want to raise chickens or horse, listen to crickets in your backyard, work in a private woodshop, or repair your own automobiles, this isn't for you



- If your lifestyle fits the constraints, condo living can be great!
  - No lawn to mow, gutters to sweep, or trash to take to the street!

Takeaways

- Our relationship with buildings has undergone tremendous evolution
  - Housing, retail, manufacturing, and office space are largely standardized
  - Standardization allows for outsourcing and sharing many aspects of buildings
  - Efficiencies and flexibility have resulted from these changes
  - *Changes in usage patterns, expectations, and the law were required*
- Cloud computing is nascent; new categories still need help
  - Great progress on Infrastructure-as-a-Service (like leasing land to build on)
  - Platform-as-a-Service is poorly defined (like apartments, condos, office & more)
- Shared buildings became successful by constraining usage and with clear laws
  - Apartments, condos, office, retail, & light manufacturing have constraints
  - Not everyone can accept the constraints… if you do, there are efficiencies
- We must define and constrain usage models for important cloud applications
  - This will allow us to offer enhanced sharing with important supporting services
- We need new laws defining rights and responsibilities for cloud computing
  - Cloud providers must have defined regulations for SLAs, privacy, and security
  - Cloud users must have defined regulations on acceptable usage and patterns of computing when using shared cloud resources