



Project Caroline: Platform as a Service

Richard Zippel, John McClain, Bob Scheifler, and Thomas Vinod Johnson

Sun Microsystems, Inc.

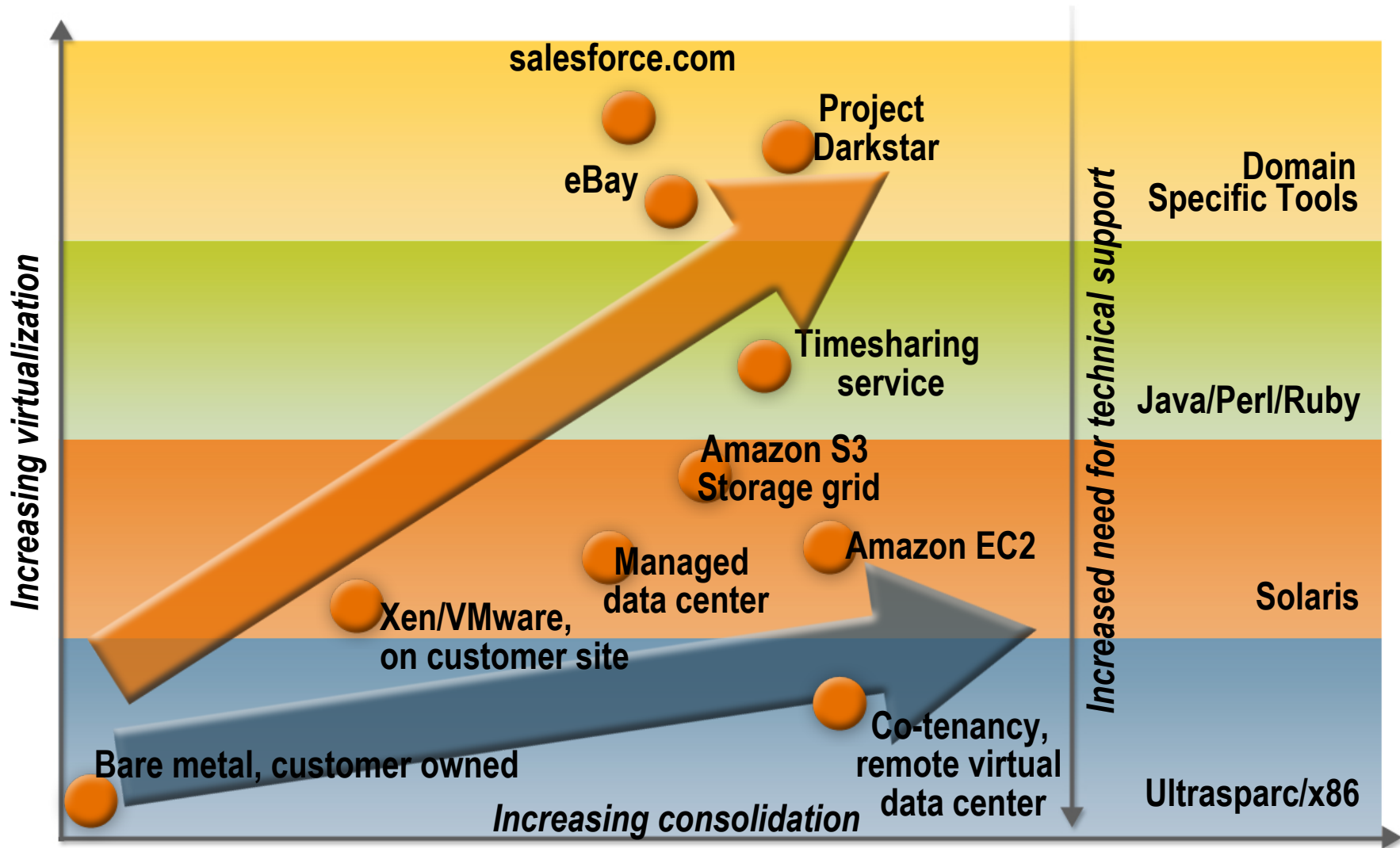
Various trademarks held by their respective owners.



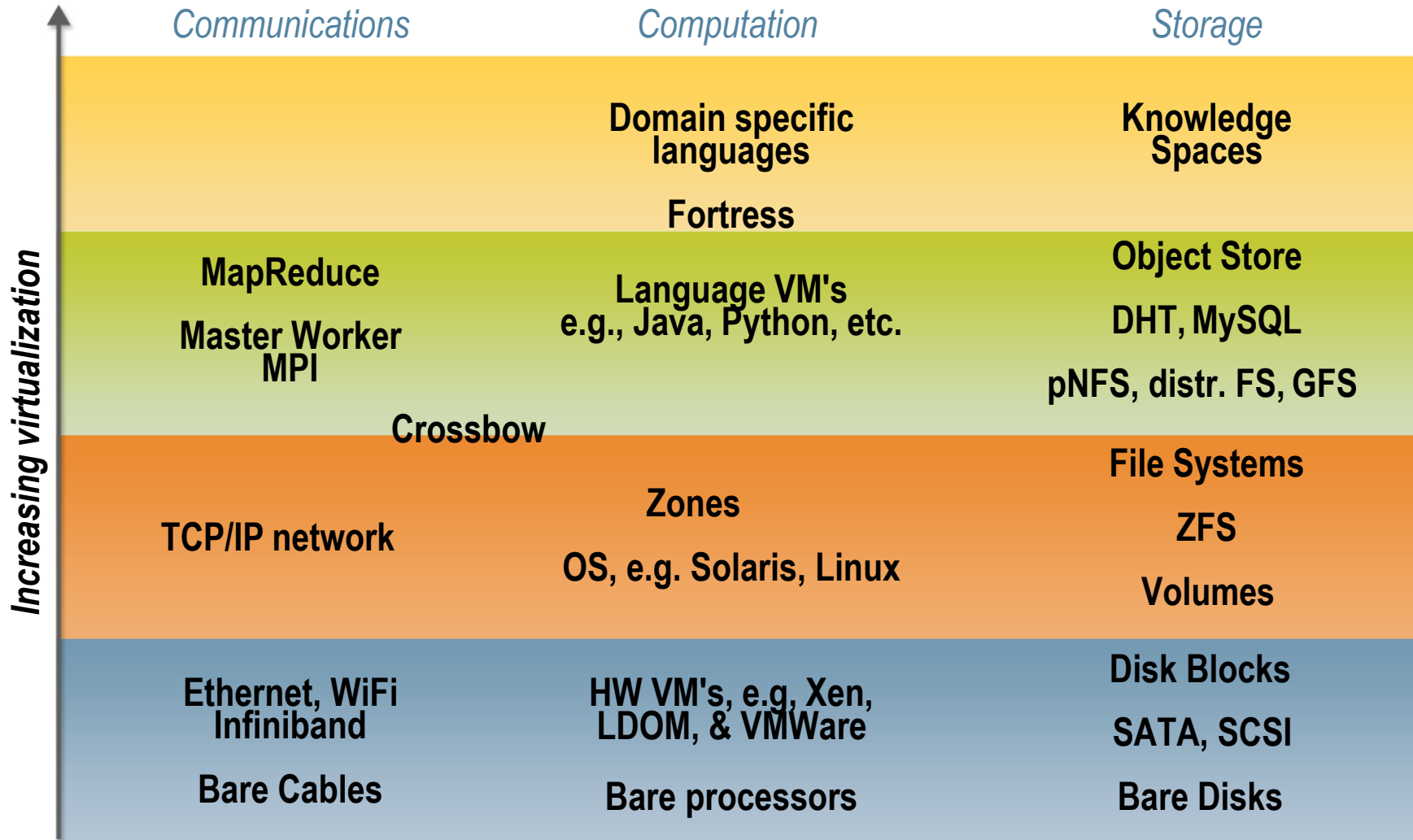
**2008
Sun Labs
Open House**



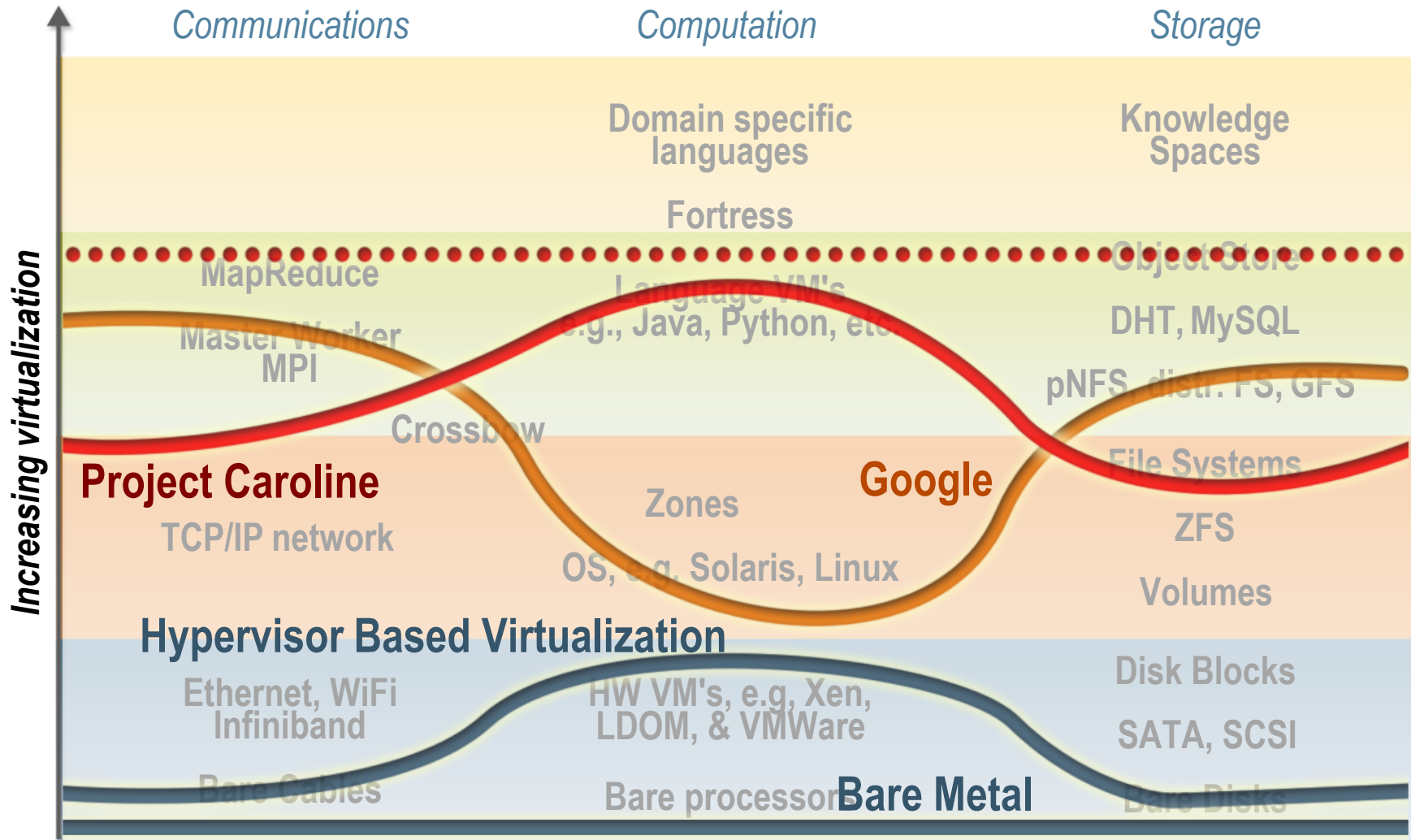
Growth and Consolidation



VirtualizationSpace



Data Center Virtualization Approaches



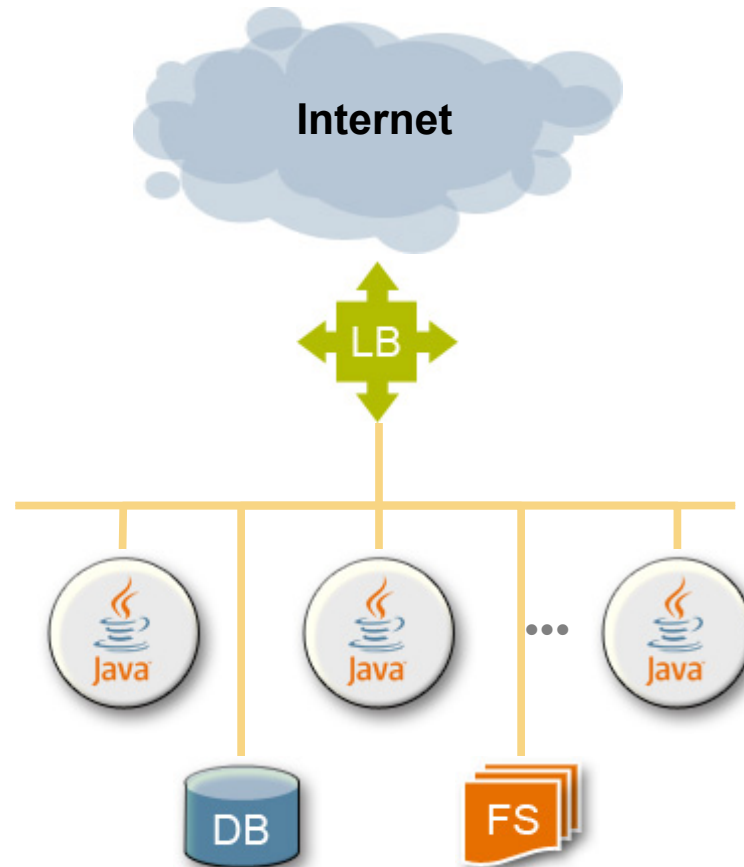
Project Caroline Team Members

- Engineers
 - > Vasiliy Baranov
 - > Frank Barnaby
 - > Pavel Bogdanov
 - > Chris Cheetham
 - > Brian Jeltema
 - > Vinod Johnson
 - > Peter Jones
 - > Ron Mann
 - > John McClain
 - > Fred Oliver
 - > Andrey Ozerov
 - > Andres Perez
 - > Juan Ramirez
 - > Robert Resendes
 - > Robert Scheifler
 - > Keith Thompson
- Staff
 - > Brendan Daly
 - > Joan MacEachern
 - > Mark Hodapp
 - > Rich Zippel

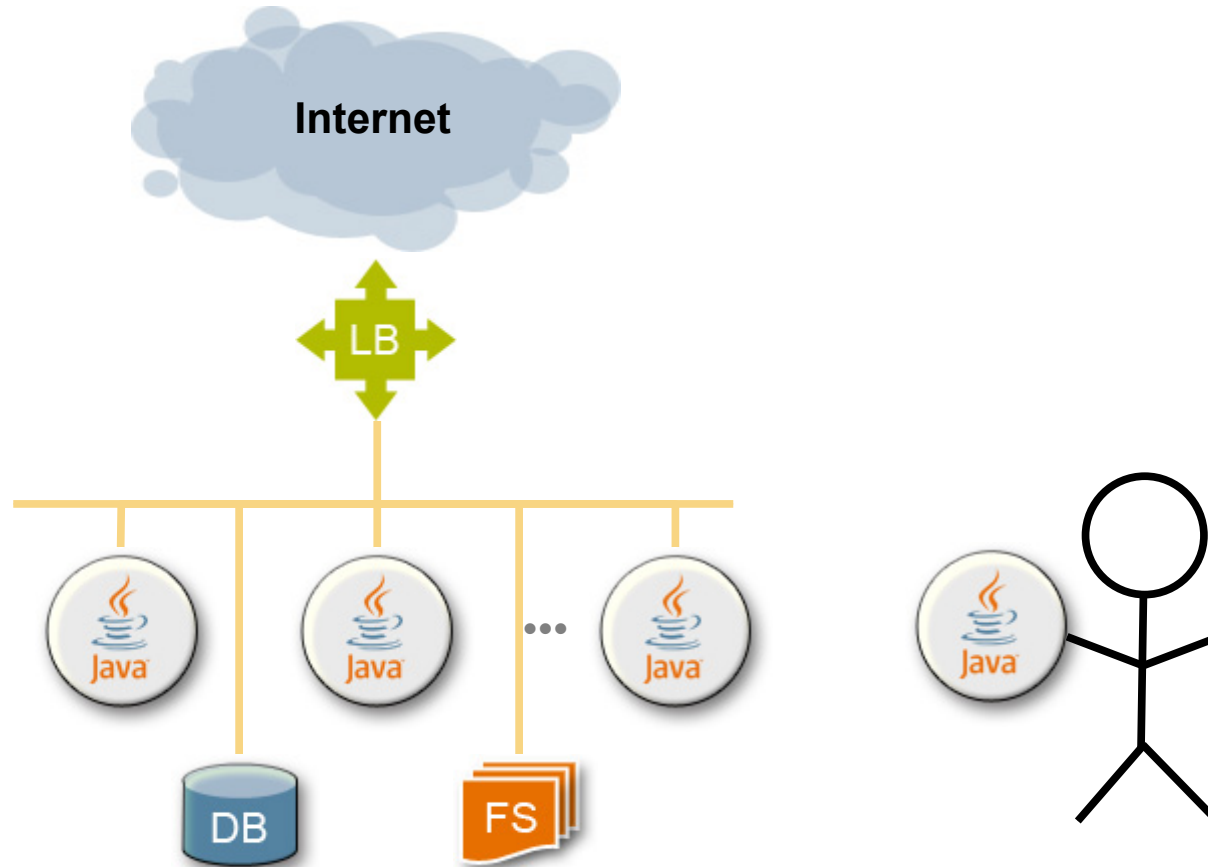
Internet Services

- All new applications will have a service component
- Move from just writing code to writing and running code
 - > Not just running, but making it accessible to millions
 - > Hard to get started
- Deployment becomes part of developer workflow
- Operations becomes key
 - > Scale, cost, uptime, change
- Can developers help operations?
- Horizontal Scale

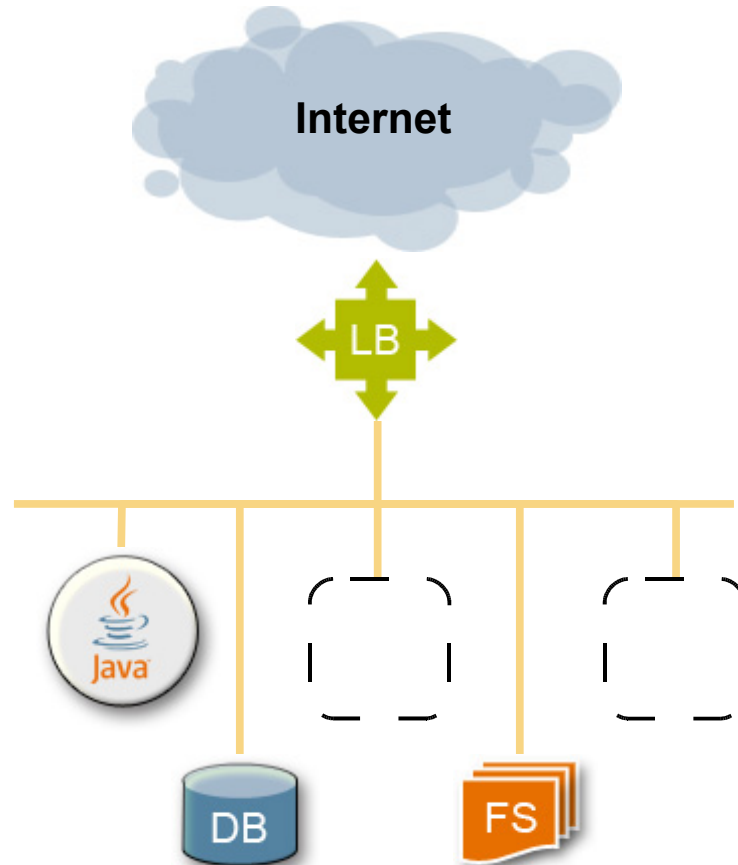
Horizontally Scaled Service



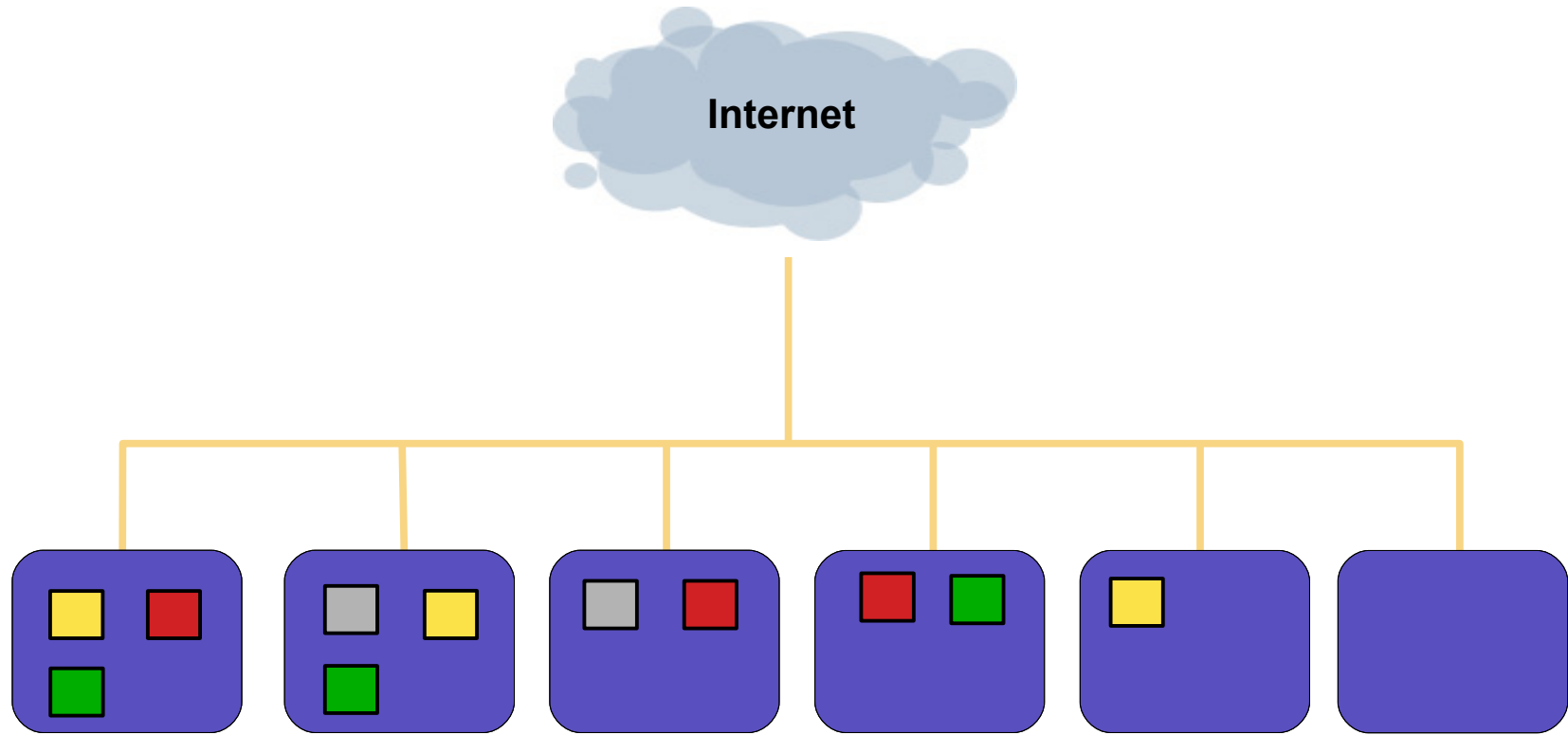
Scaling Up



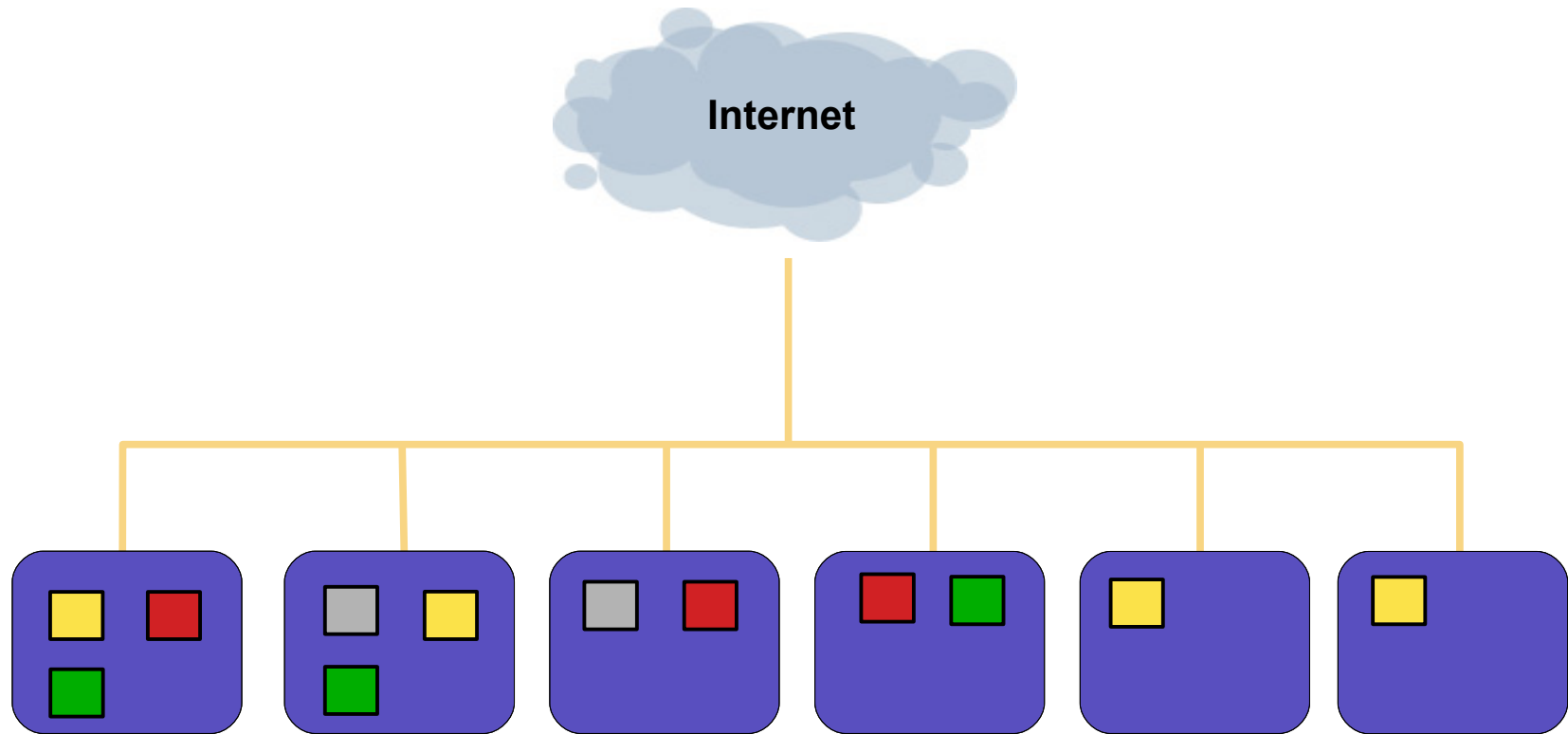
Scaling Down



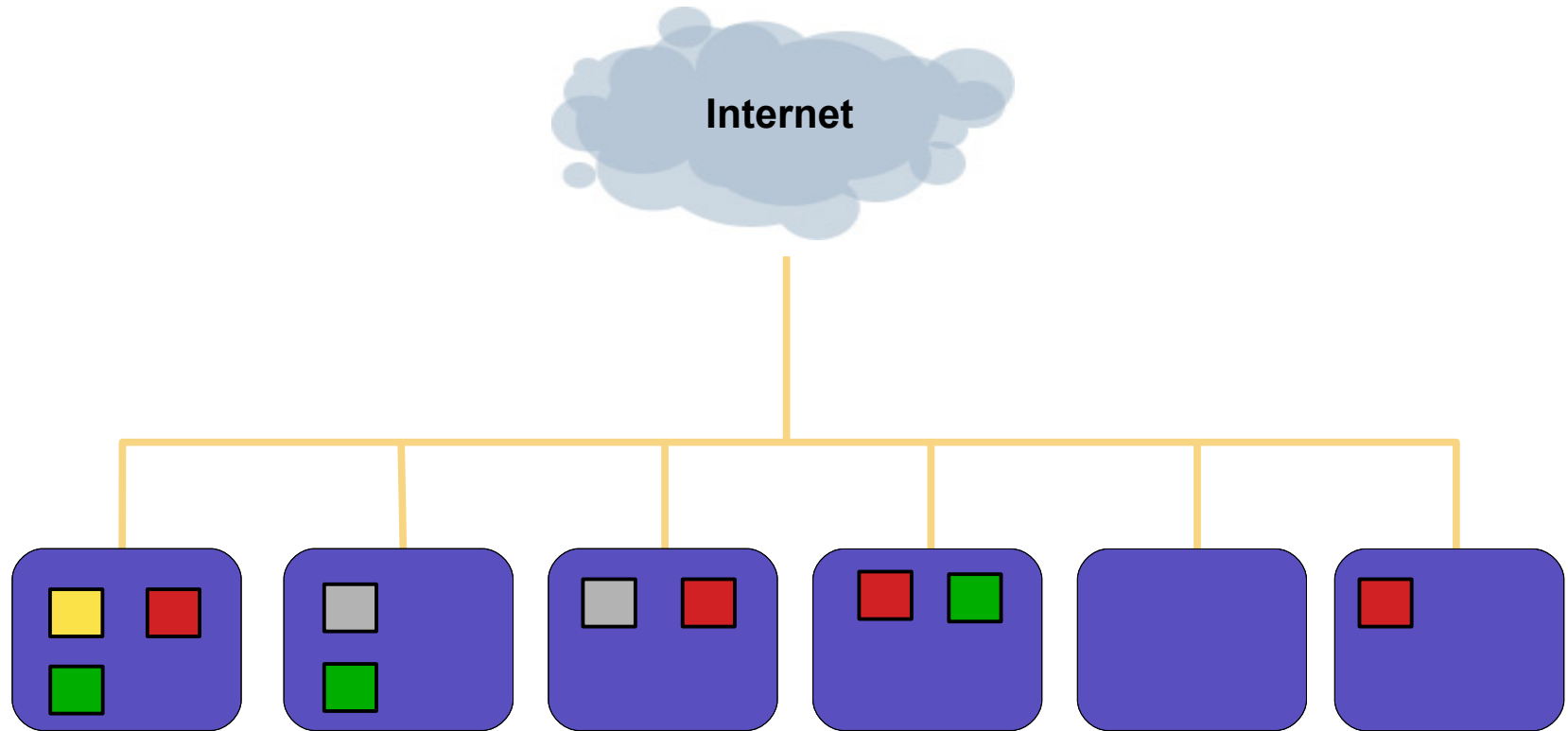
Utility Computing



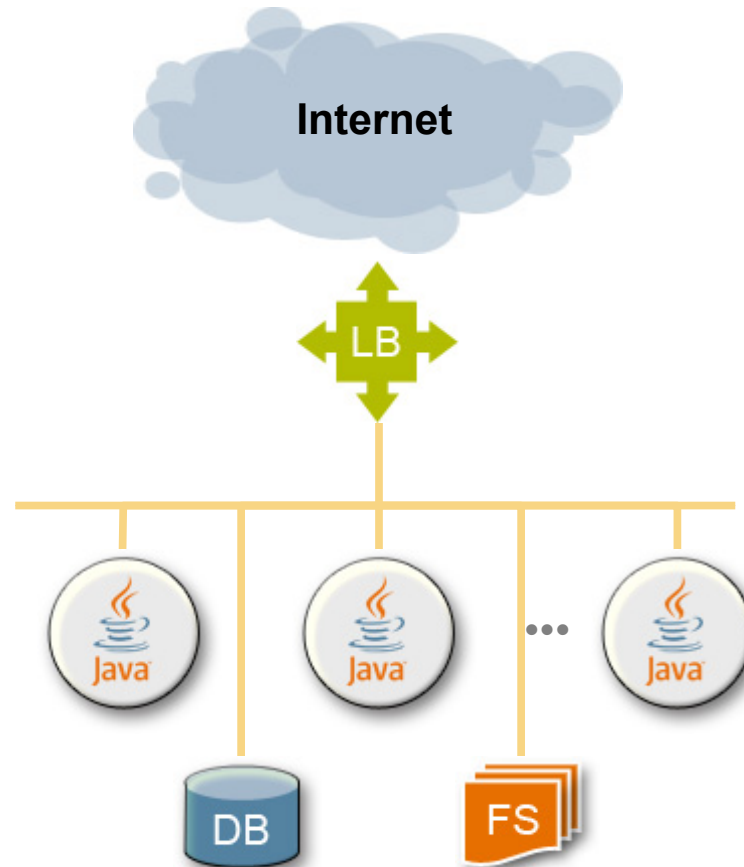
Scaling Up



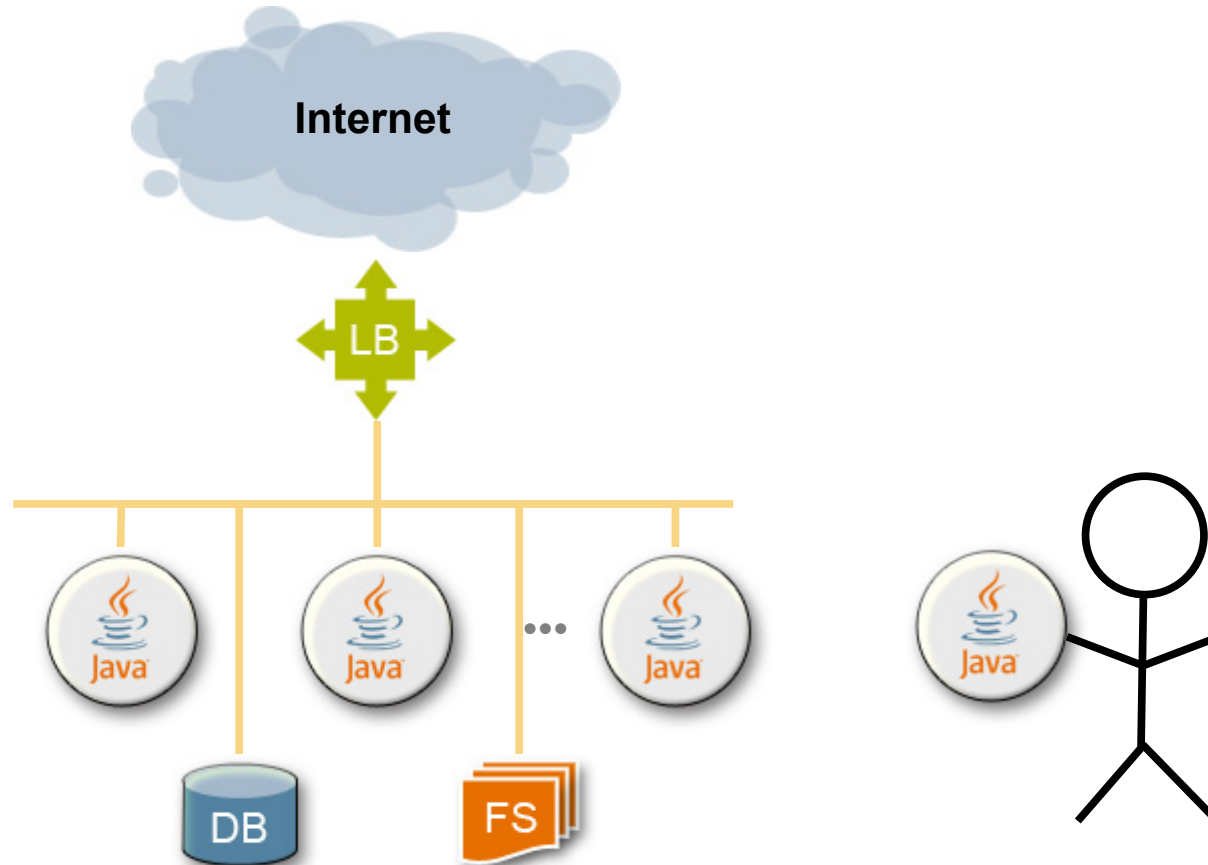
Scaling Down



Need to Maintain Logical View



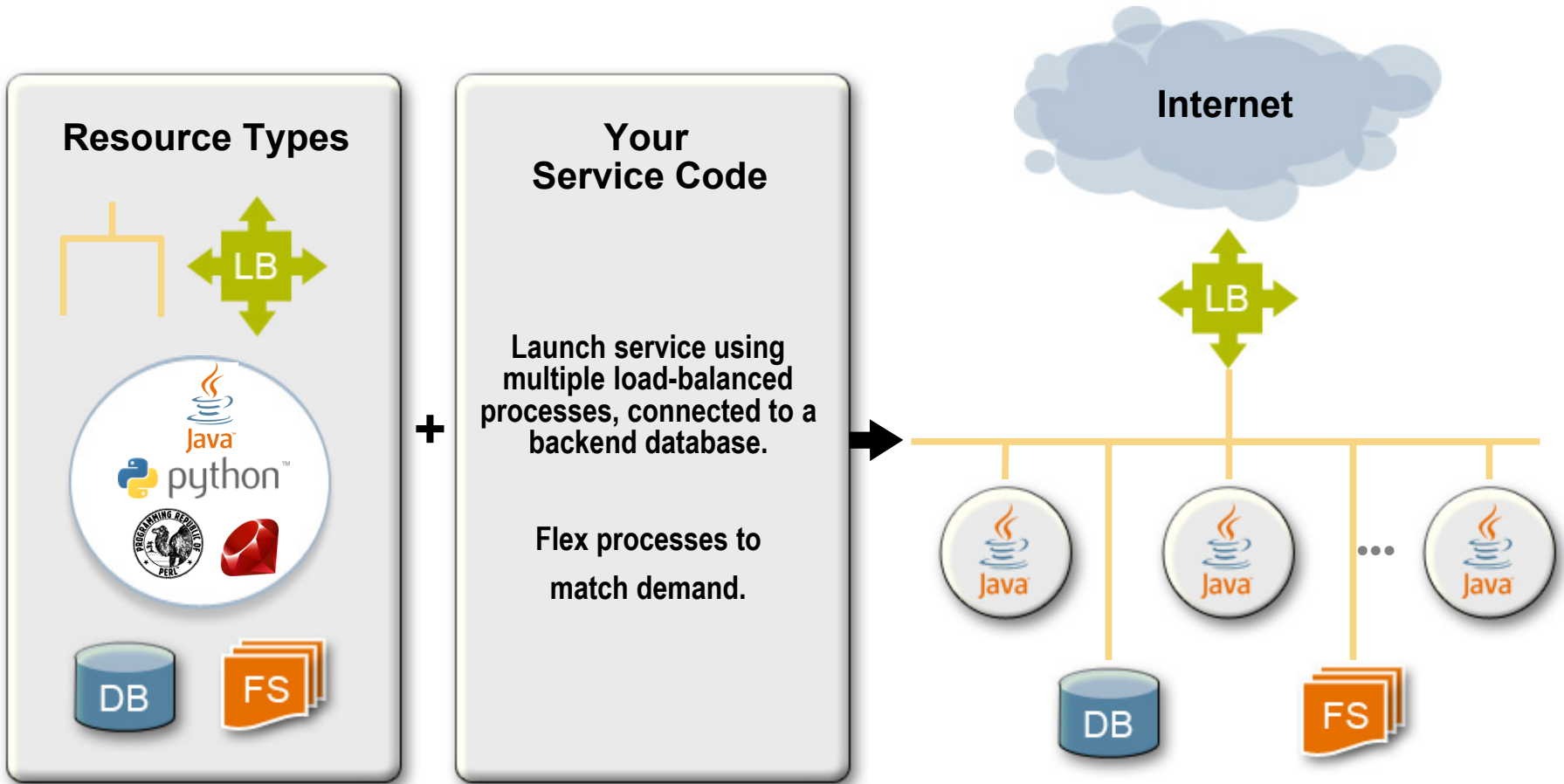
Keep the Carbon Units?



What is Project Caroline

- Research project developing a **platform** for development and deployment of long-running Internet services
- **Utility scale**: lots of customers and services on a single large shared grid, with **secure isolation**
- Full **programmatic** control of **distributed** compute, storage, and network resources
- Services can configure and **flex** their own resource usage up and down in real time
- High level of resource abstraction

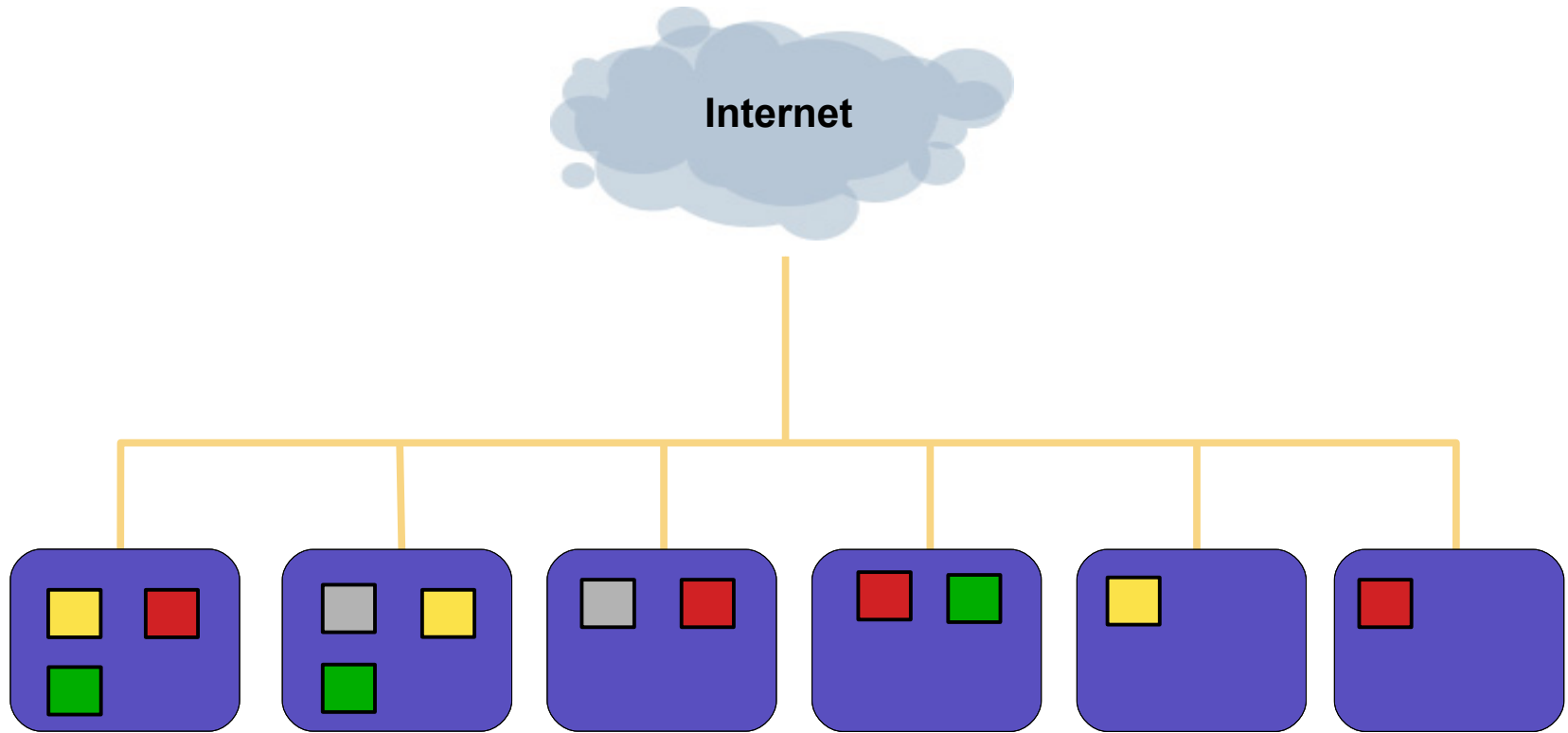
Developer View



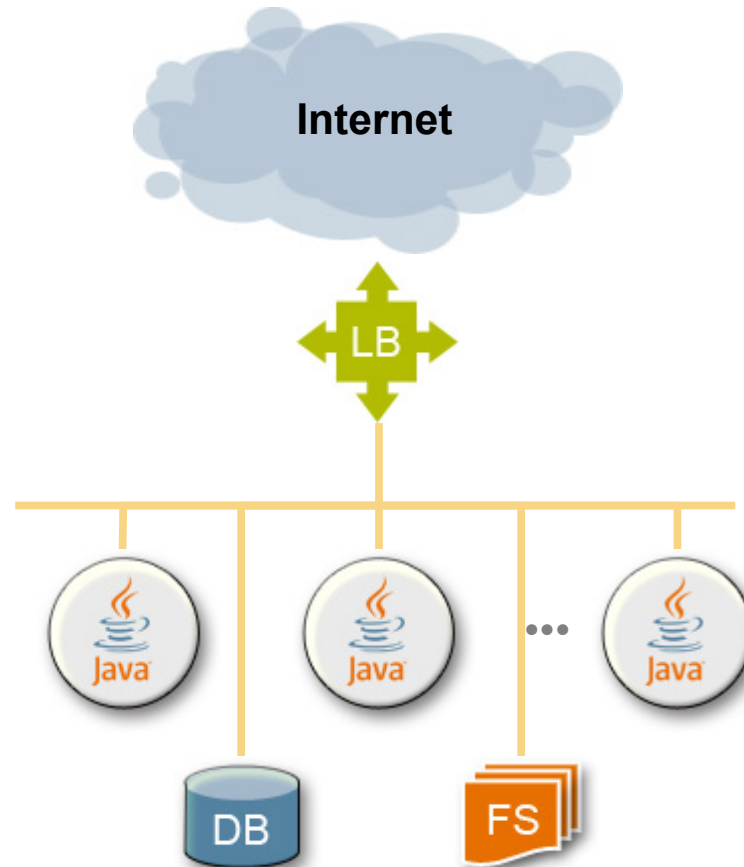
Target Applications

- Long running Internet Services
 - > In particular Software as a Service offerings
- Expressed in high level languages
 - > OS/instruction set independent process model
 - > Java, Perl, Python, etc.
- Project Caroline is a new platform, implies new applications
 - > Should not need to start from scratch though
- Horizontal scale
- Want to vary their distributed resource usage over time

How Do We Get from Here :



To Here?



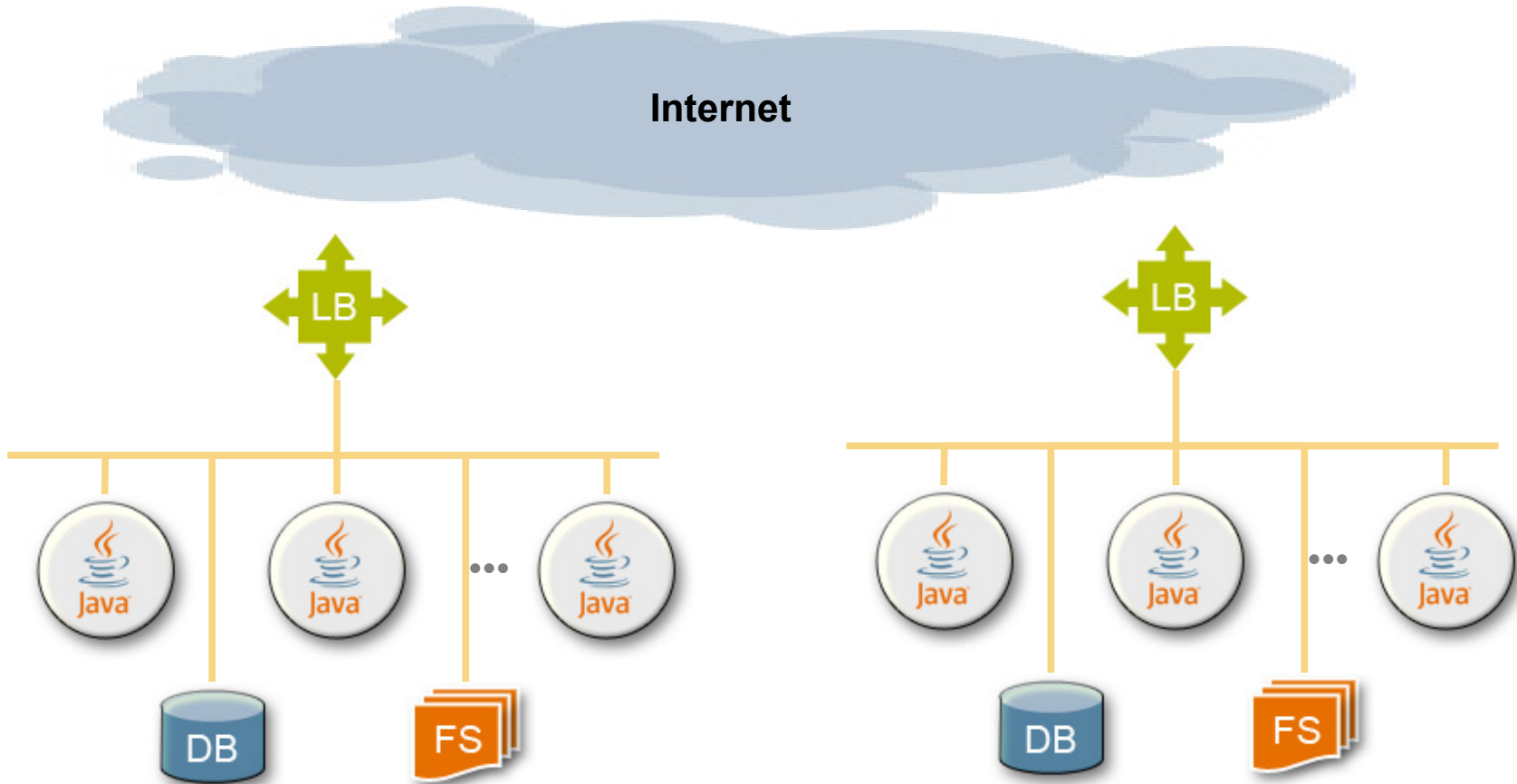
High Level Virtualization

- Resources are exposed through high level abstractions
 - > Hardware/OS/location independent programs
 - > IP sub-nets
 - > Network accessible file systems and database
- High isolation with low overhead
 - > Faster allocation times
- Improves developer productivity
 - > Less to specify to create a resource
 - > Allows for finer grain sizes

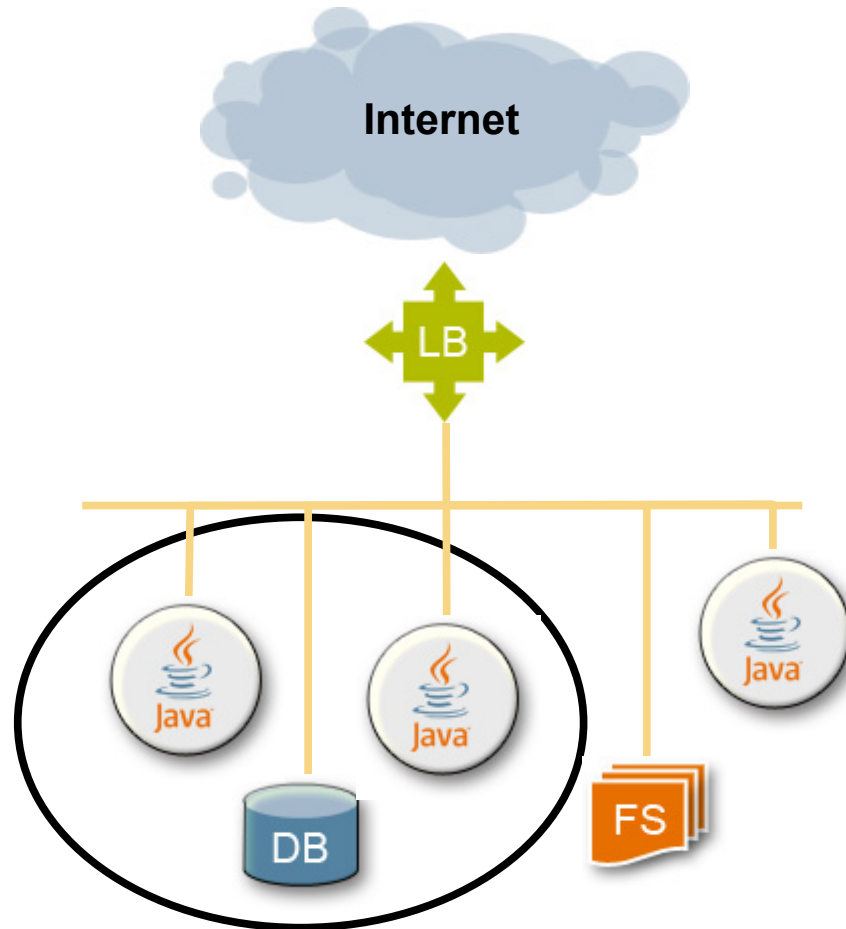
Impact of Project Caroline Model

- Automate deployment & day-to-day operations
 - > Faster response times
 - > Capture knowledge in programs not process books
- Services construct their environment instead of being inserted into an existing one
 - > Simpler for operations and developers
- Isolation between service instances
- Multi-process components
- Developer workflow

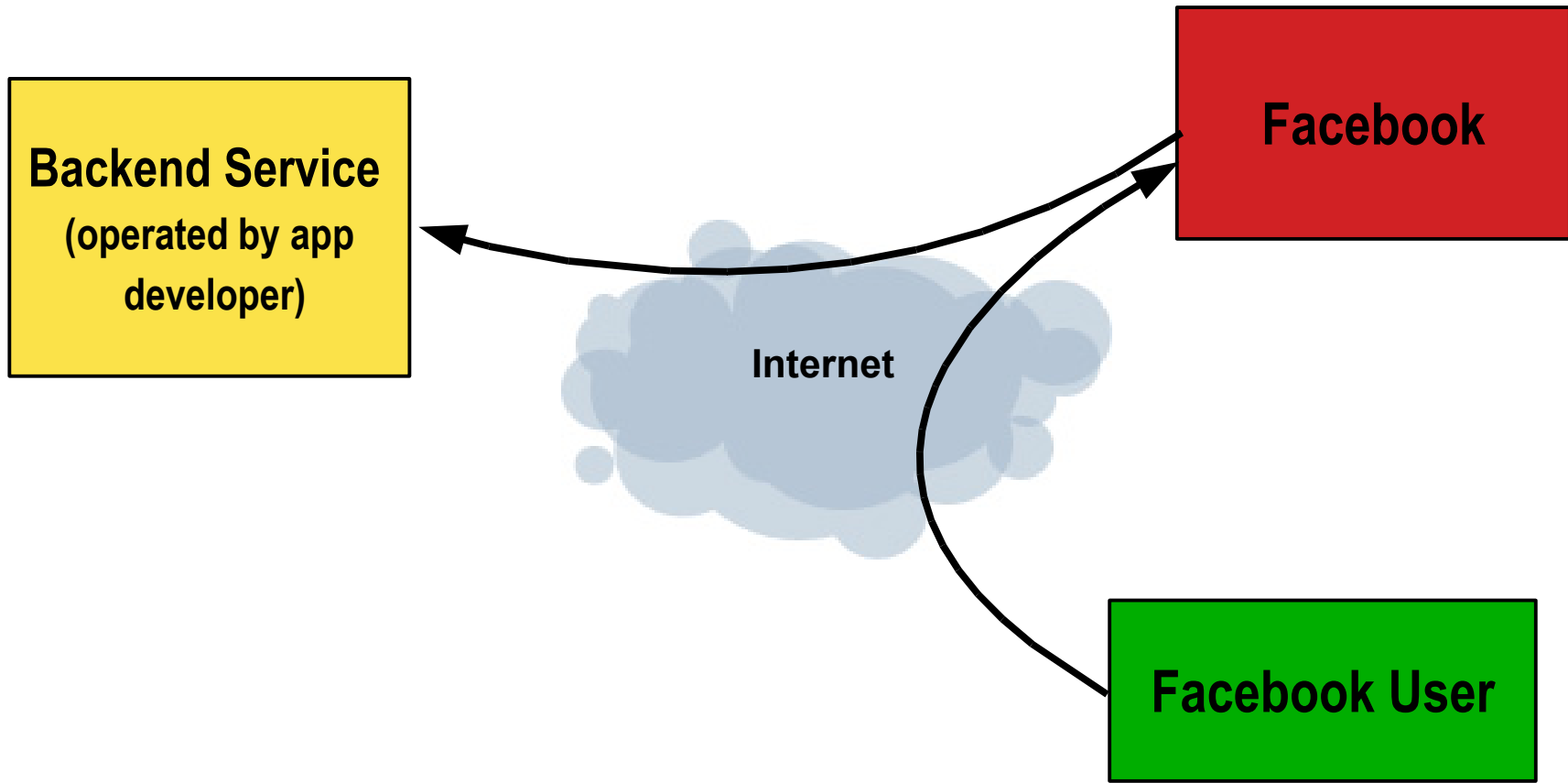
Deploy the Application Twice



Macro Components



Facebook Application Demo



Facebook Application Demo

**Grid
(Burlington MA)**

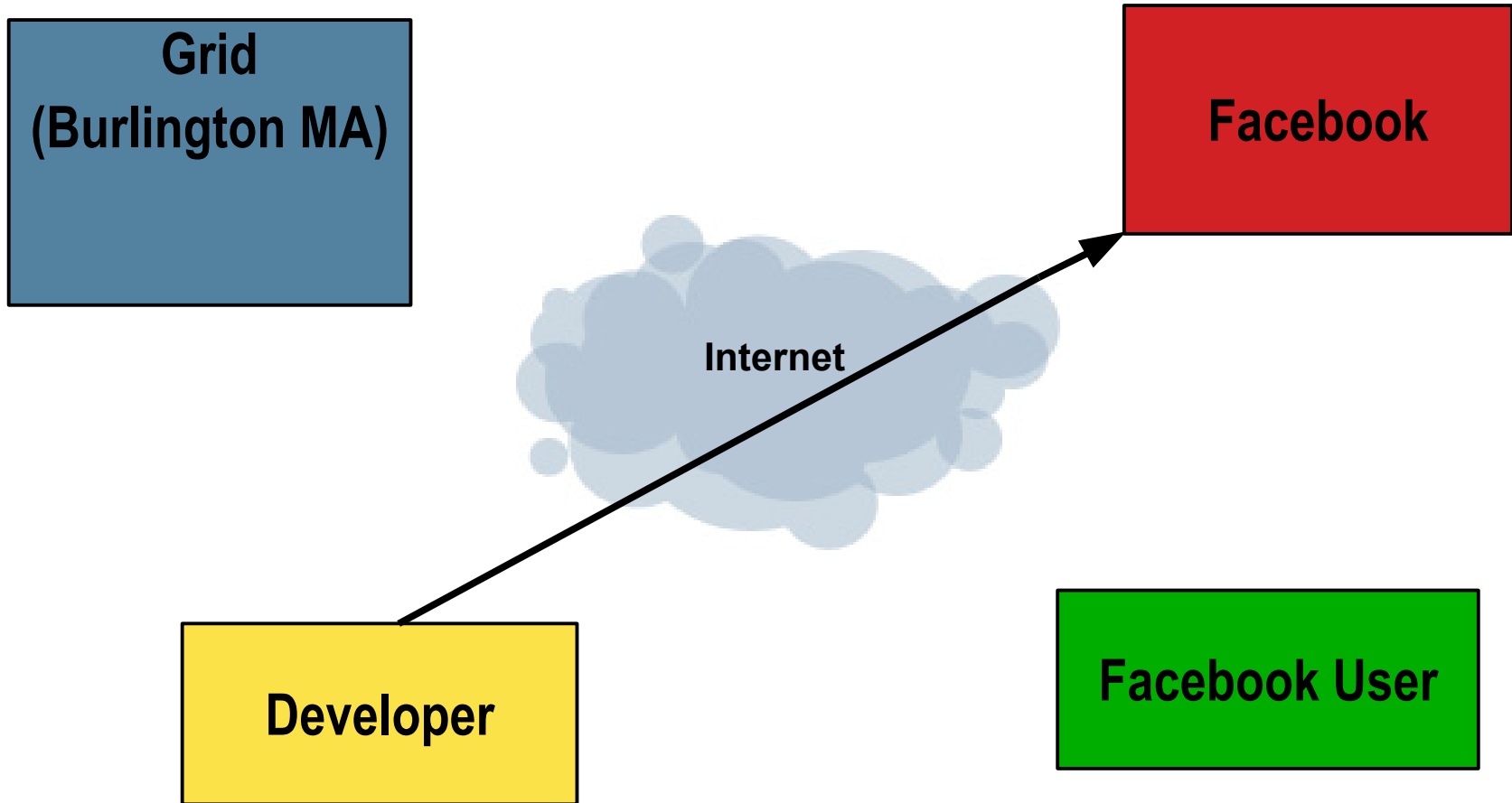
Facebook

Internet

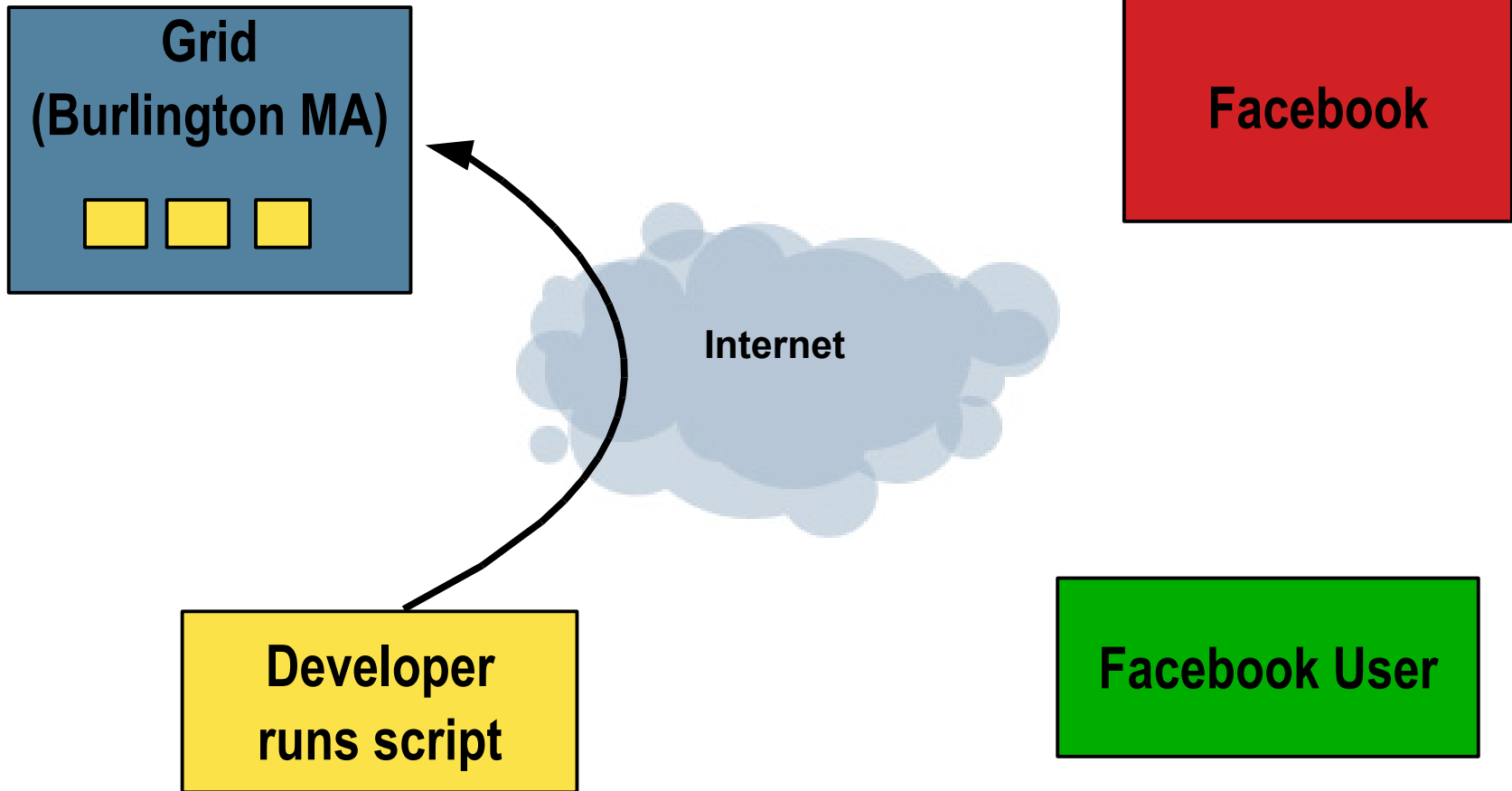
Developer

Facebook User

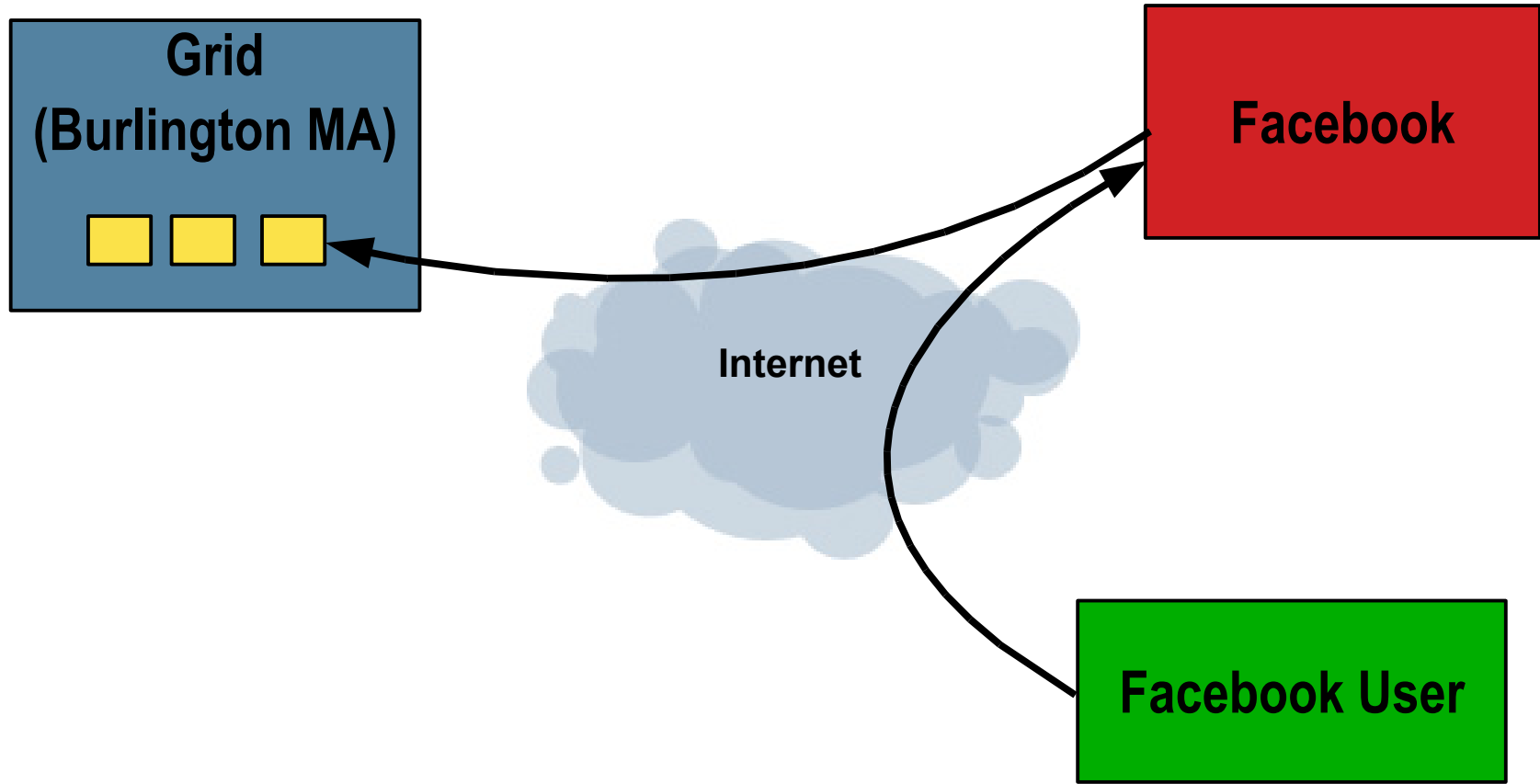
Facebook Application Demo



Facebook Application Demo



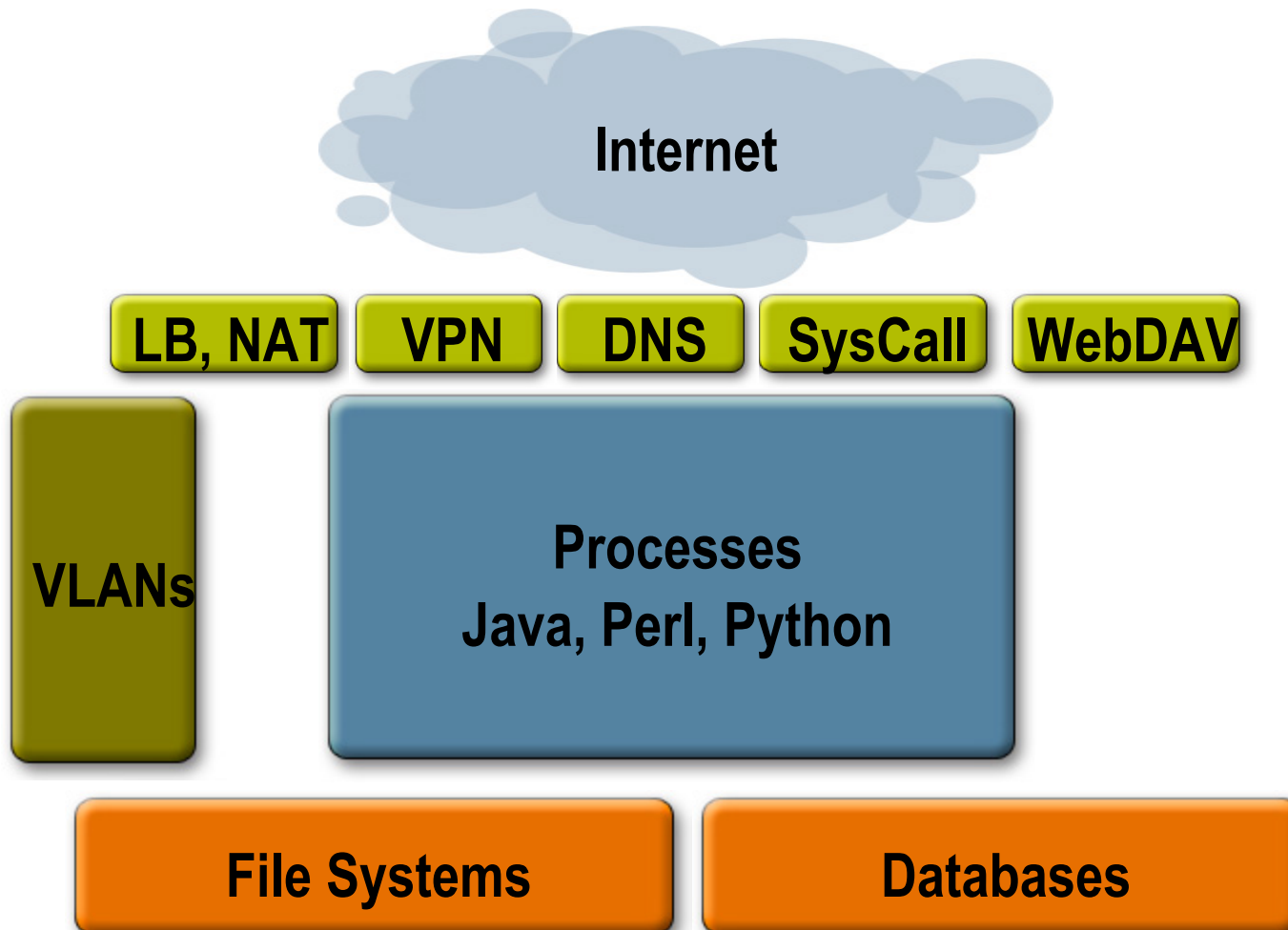
Facebook Application Demo



Platform Resources

- Process (Java, Perl, Python)
- ZFS file system (access via NFS and WebDAV)
- PostgreSQL database
- Network (VLAN)
- IP address (internal, Internet)
- Internet connectivity (Load Balancer, NAT, VPN)
- DNS record (host name \leftrightarrow IP address)

Platform View



Platform Isolation

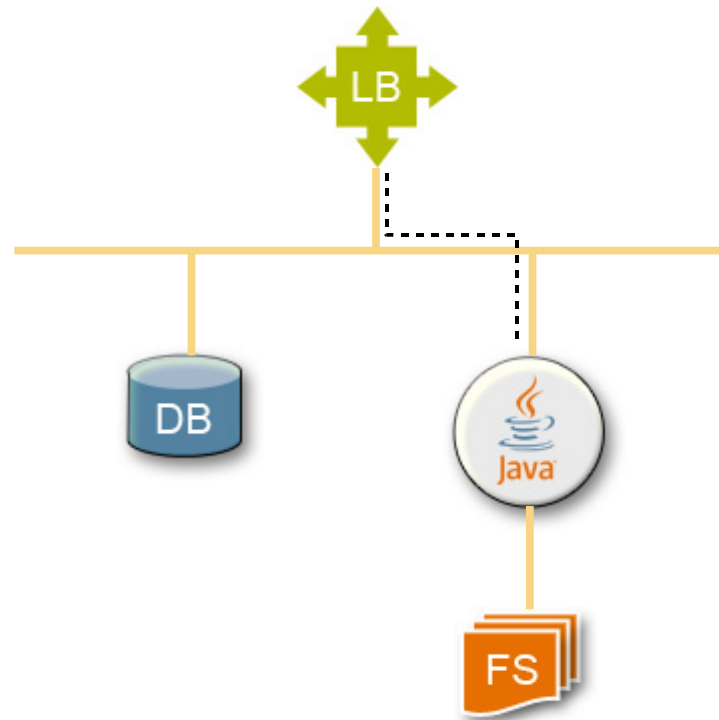
- No access to resources of another grid account
 - > Except through explicit Internet connectivity
- No connectivity between internal networks
 - > Even within a single grid account
 - > Except through explicit Internet connectivity

Basic Resource Operations

- **CRUD**
 - > Create (name, configuration)
 - System assigns a UUID
 - > Read configuration, state, and metrics
 - Lookup by name, UUID
 - > Update configuration
 - > Delete
- **Events**
 - > On create, update, delete, state change

Example

<http://www.mydomain>



File System Creation

- Configuration
 - > Quota
 - > Reservation
 - > Read-only

```
myFS = grid.createBaseFileSystem("myFS",  
                                new BaseFileSystemConfiguration());
```

Network Creation

- Configuration
 - > Number of IP addresses
 - > Metadata

```
myNet = grid.createNetwork("myNet", 16,  
                           new CustomerNetworkConfiguration());
```

IP Address Allocation

- No configuration

```
dbAddr = myNet.allocateAddress("dbAddr");
```

```
intAddr = myNet.allocateAddress("intAddr");
```

```
extAddr = grid.allocateExternalAddress("extAddr");
```

Database Creation

- Configuration
 - > Database IP address
 - > Allowed client IP addresses

```
myDB = grid.createPostgreSQLDatabase("myDB",  
    new PostgreSQLConfiguration(  
        dbAddr.getUUID(), null));
```

Process Creation

- Configuration
 - > Command line (argv[0] is an enum: Java, Perl, Python)
 - > File system mounts
 - > IP addresses
 - > Exit action (park, restart, destroy)
 - > Input and output files (standard input, output, error)
 - > Working and home directories
 - > Packet filter rules (firewall)
 - > Environment variables
 - > Location constraints
 - > Metadata

Process Creation

```
pcfg = new ProcessConfiguration();
pcfg.setRuntimeEnvironment(RuntimeEnvironment.JAVA);
pcfg.setCommandLine(new String[]{"-jar", "server.jar"});
pcfg.setFileSystems(Arrays.asList(
    new FileSystemMountParameters(myFS.getUUID(), "home")));
pcfg.setWorkingDirectory("/files/home");
pcfg.setNetworkAddresses(Arrays.asList(intAddr.getUUID()));
pcfg.setSystemSinks("out-and-err.txt", false);
myProc = grid.createProcessRegistration("myProc", pcfg);
myProc.start(false);
```

JRuby Version

```
myFS = create_file_system "myFS"  
myNet = create_network "myNet", 16  
intAddr = myNet.allocate_address "intAddr"  
pcfg = ProcessConfiguration.new  
pcfg.runtime_environment = RuntimeEnvironment::JAVA  
pcfg.command_line = ["-jar", "server.jar"]  
pcfg.add_file_system "myFS", "home"  
pcfg.working_directory = "/files/home"  
pcfg.add_address "myNet", "intAddr"  
pcfg.set_system_sinks "out-and-err.txt", false  
myProc = create_process_registration "myProc", pcfg  
myProc.start false
```

Layer 4 Load Balancer Creation

- Configuration
 - > External IP address, port, and protocol
 - > Set of: <internal IP address, port, and weight>
 - > Connection timeout

```
scfg = new L4VirtualServiceConfiguration();
scfg.setExternalNetworkAddress(extAddr.getUUID());
scfg.setPort(80);
scfg.setProtocol(Protocol.TCP);
scfg.setRealServices(Arrays.asList(
    new RealService(intAddr.getUUID(), 8080)));
myLB = grid.createNetworkSetting("myLB", scfg);
```

DNS Record Creation

- Configuration
 - > Host name
 - > Collection of IP addresses
 - > Time to live (TTL)

```
bcfg = new HostNameBindingConfiguration();  
bcfg.setHostName("www");  
bcfg.setAddresses(Arrays.asList(extAddr.getUUID()));  
myDNS = grid.getExternalHostNameZone().createBinding(  
    "myDNS", bcfg);
```

Tearing Down

```
myDNS.destroy();  
myLB.delete();  
myProc.destroy(Long.MAX_VALUE);  
myDB.destroy();  
extAddr.delete();  
intAddr.delete();  
dbAddr.delete();  
myNet.delete();  
myFS.destroy();
```

Other Process Features

- Self-contained node-independent configuration
- State
 - > Run state
 - > Incarnation number
 - > Last incarnation outcome (exit value, signal, exception)
 - > Process-is-stuck flag
- Per-process and per-thread execution metrics
- Dynamic file system mounts
- Native libraries not supported
- No local process fork or exec

Other File System Features

- Automatic WebDAV access from the Internet
 - > HTTPS with authentication
- Read-only snapshots
 - > Shares disk space with original
- Rollback to most recent snapshot
- Copy-on-write clones
 - > Shares disk space with snapshot

Other Internet Connectivity

- Inbound & outbound traffic
 - > Process direct binding to Internet address
 - > Static NAT (Internet address \leftrightarrow internal address)
 - > VPN (bind off-grid process to internal address)
- Inbound traffic
 - > Layer 4 load balancing: TCP, SSL, UDP
 - Internet address+port \rightarrow {internal address+port}
 - > Layer 7 load balancing: HTTP, HTTPS
 - Internet address+port+{URI filter \rightarrow {internal address+port}}
- Outbound traffic
 - > Dynamic NAT (internal address \rightarrow Internet address)

Other DNS Features

- Internet-facing public DNS zone
 - > Delegated domain from external DNS server/registrars
 - > Domain name is attribute of grid account
- Grid-internal private DNS zone
 - > *accountname.caroline.local*

Horizontal Scale

- It is a design and development time concern

Achieving horizontal scale

- Partitioning the problem
- Determine the process model
- Communication overhead
- One or more pieces of a large system are typically down
- Rethink functionality
- Relax constraints

Project Caroline's process model

- Partitioning is light weight, fine grained
- Retains high levels of isolation
- Macro versus micro control in operating systems
- Developer efficiency
- Operator efficiency
 - > “Fail-in-place” economics
- One model to scale up or down
 - > Accrue the benefits of “fail-in-place” economics

Deploying the service

- Consequence of designing for horizontal scale
 - > Increasingly distributed, moving parts
 - > Internal application architecture becomes a deployer concern
- Software as a service
 - > Deployment is no longer somebody else's problem
 - > Services can and are delivered by self contained teams
 - May be in concert with other teams

Deploying the service (continued)

- Process books don't help
- Not a simple matter of following the instructions
 - > Vague, out of date, incomplete
- Developer plays a variety of roles
 - > Set up the network, IP addresses
 - > Provision and deploy software images
 - > Configure descriptors

Running the service

- What processes need to be running, scheduled, stopped?
 - > Process failure, OS failure, node failure
- Scaling the service
 - > Performance monitoring
 - > Acquiring or releasing resources
 - > Provisioning them into and out of the service
- Upgrades
 - > Continuous improvement, testing, roll out

Running the service (continued)

- Exploiting the utility model
 - > Just in time usage of resources
 - > Makes agility of deployment more important
- Complexity in deployment
 - > Adds to delays
 - > Adds to errors
 - > Adds to cost
- Need unified set of tools for provisioning and runtime management

Solving the deployment problem

- How does a process deploy itself?
 - > Hit run
 - Call malloc, spawn a new thread, exec a new process
- How does the horizontally scaled application do it?
 - > Project Caroline APIs
 - > Resource management is conceptually light weight
 - > Events to track changes
 - > Run states to manage process states

Solving the deployment problem

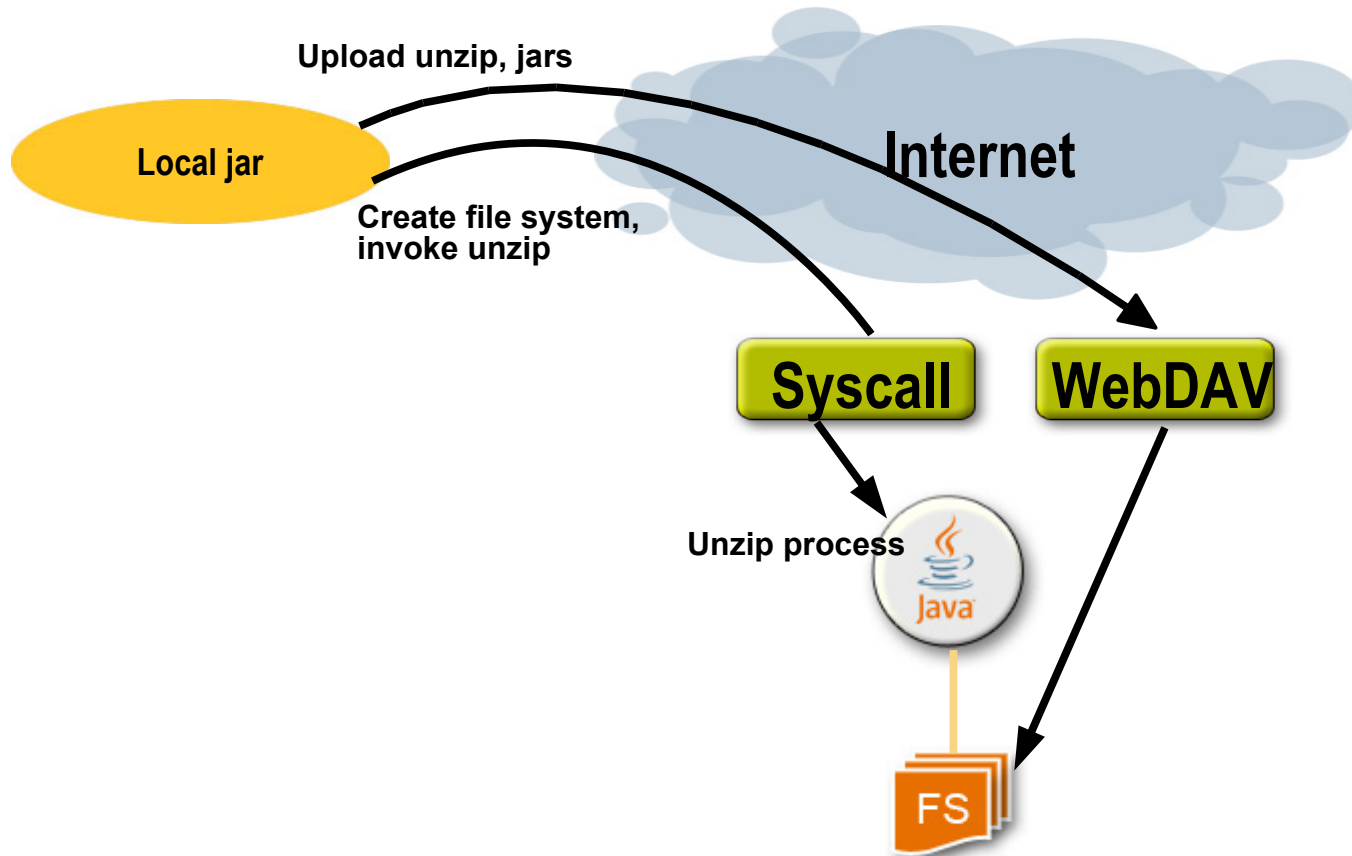
- Applications can participate in their own operation
 - > A process book that a machine understands
 - > Can go back to hiding their architecture
 - > Can better exploit the utility model
- Platform enables reuse
 - > Leverage higher levels or abstraction (macro components) instead of solving all problems
 - > Self managing components
 - > Can plug in to higher level policies

Project Caroline Web Server

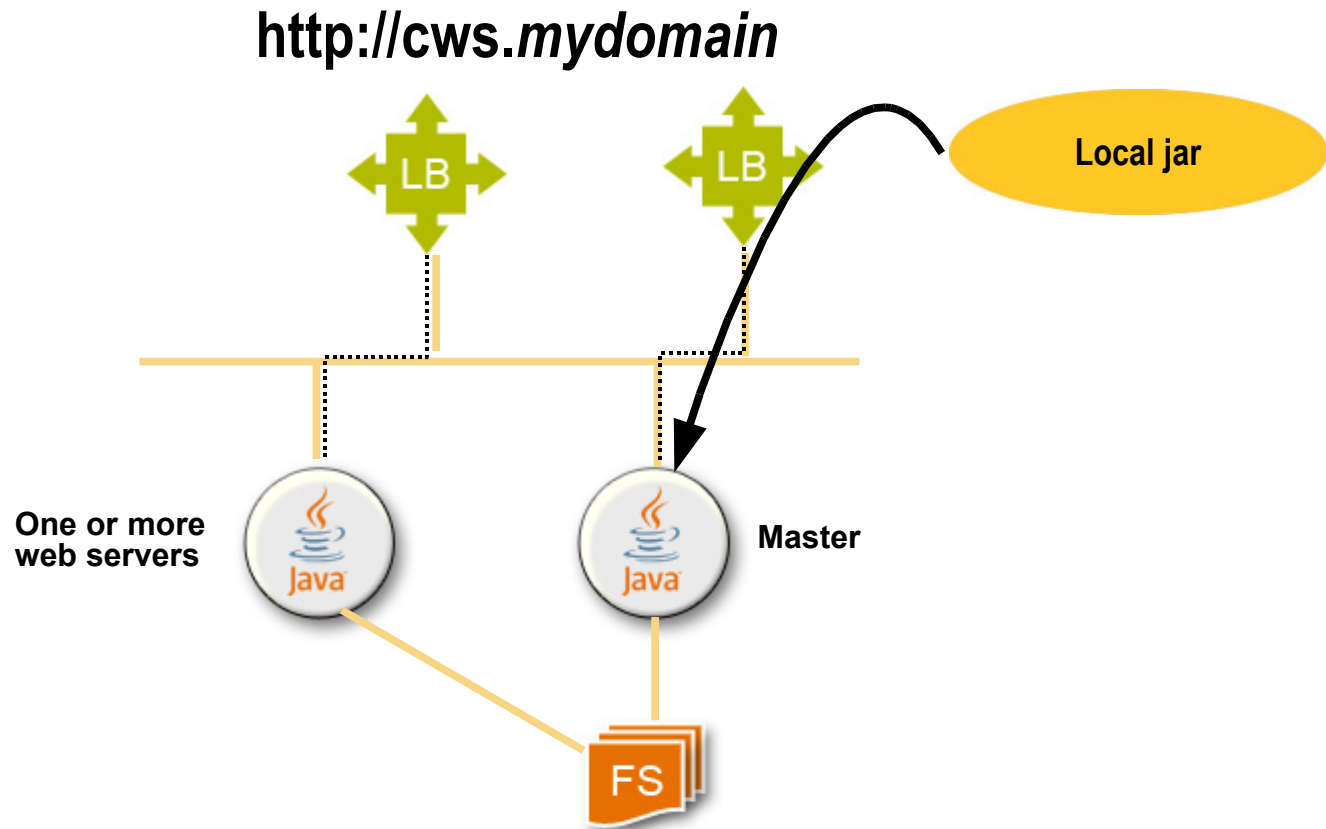
- Example of higher level component
 - > Leverages light weight process model for scale
 - > Leverages API for self management
- An instantiation of a horizontally scaled Web tier
- Built from Project Glassfish v3
- Self install, deploy, uninstall
- Handles multiple instances
- Self managing for handling variation in load

Demo

Install



Start, flex, stop



Tools

- NetBeans Plug-in
- Grid Accessor GUI
- Apache Ant Tasks
- JRuby binding of platform API
- JRuby centric tool chain
- JRuby based interactive shell
- Tunnel
- Unzip
- All tools layered on top of platform API

Status

- PE grid went live on Internet in March
 - > Feedback on model
 - > Experience with grid operations
- Project Aura hosting blog recommendation service on PE grid
- First non-Sun research partner given access in March
 - > Will be adding additional external research partners

www.projectcaroline.net

- Documentation & examples
- Forums
- Tool and component downloads
- Source code (GPLv2)
- Hosted and developed on grid in customer account
 - > Drupal on Quercus on Apache Tomcat
 - > PostgreSQL
 - > Apache James
 - > Deployed by Apache Ant Script
 - zero to web site in 6 minutes



Project Caroline: Platform as a Service

Richard Zippel, John McClain, Bob Scheifler, and Thomas Vinod Johnson

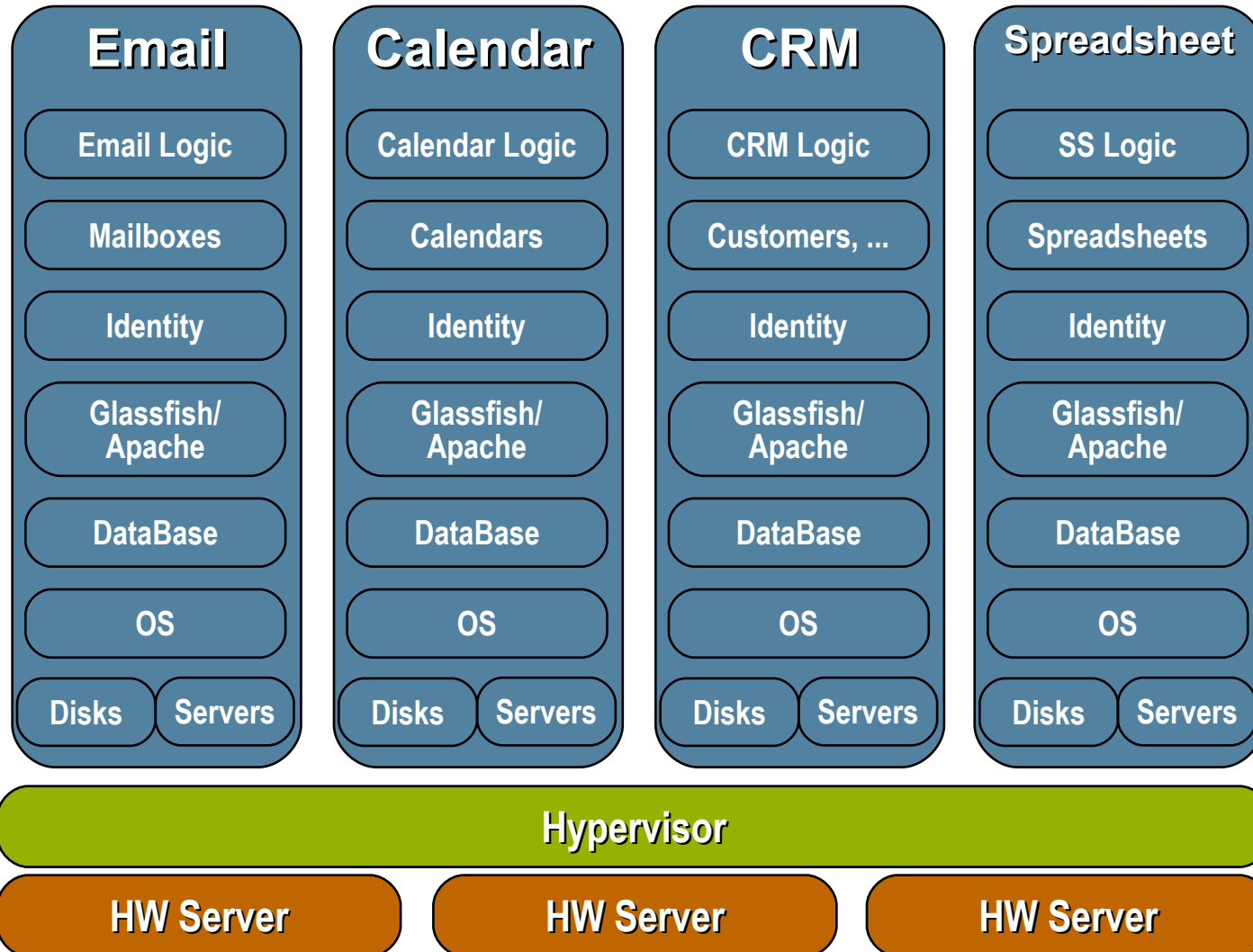
Sun Microsystems, Inc.



**2008
Sun Labs
Open House**



Typical Infrastructure Today



Cloud Infrastructure

