The Océano Project

Intelligent eUtilities Infrastructure

Towards Self-Managed Server Farms

Germán Goldszmidt
(gsg@us.ibm.com)
and The Océano Team

IBM Research

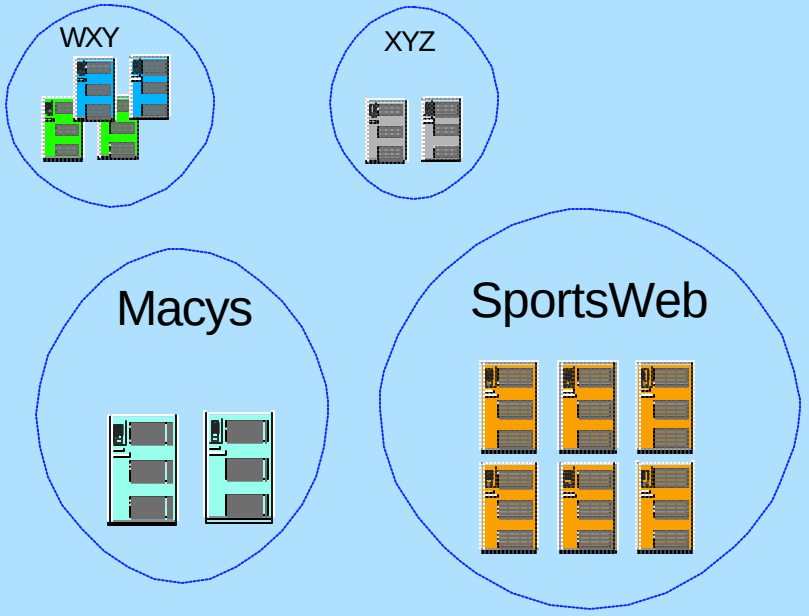
October 2001

Océano - Presentation Outline

- Motivation
- Sample Scenario
- Architecture
- Components
- Status

Multi-Customer Farms: Today

- Independent Islands

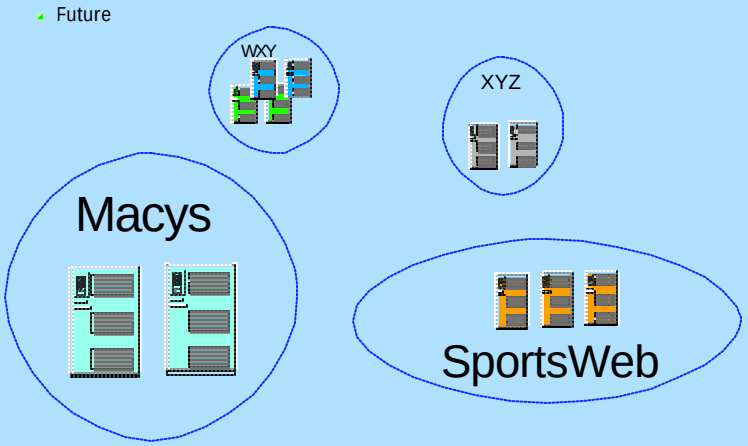


Today

Problems

- Non-shared dedicated hardware
  - for each customer/domain
- Over provisioning
  - (peak loads 10:1)
- Lack rapid response to demand
- **TCO high**

Océano Farms: Future



Virtualize the hardware
Unified management

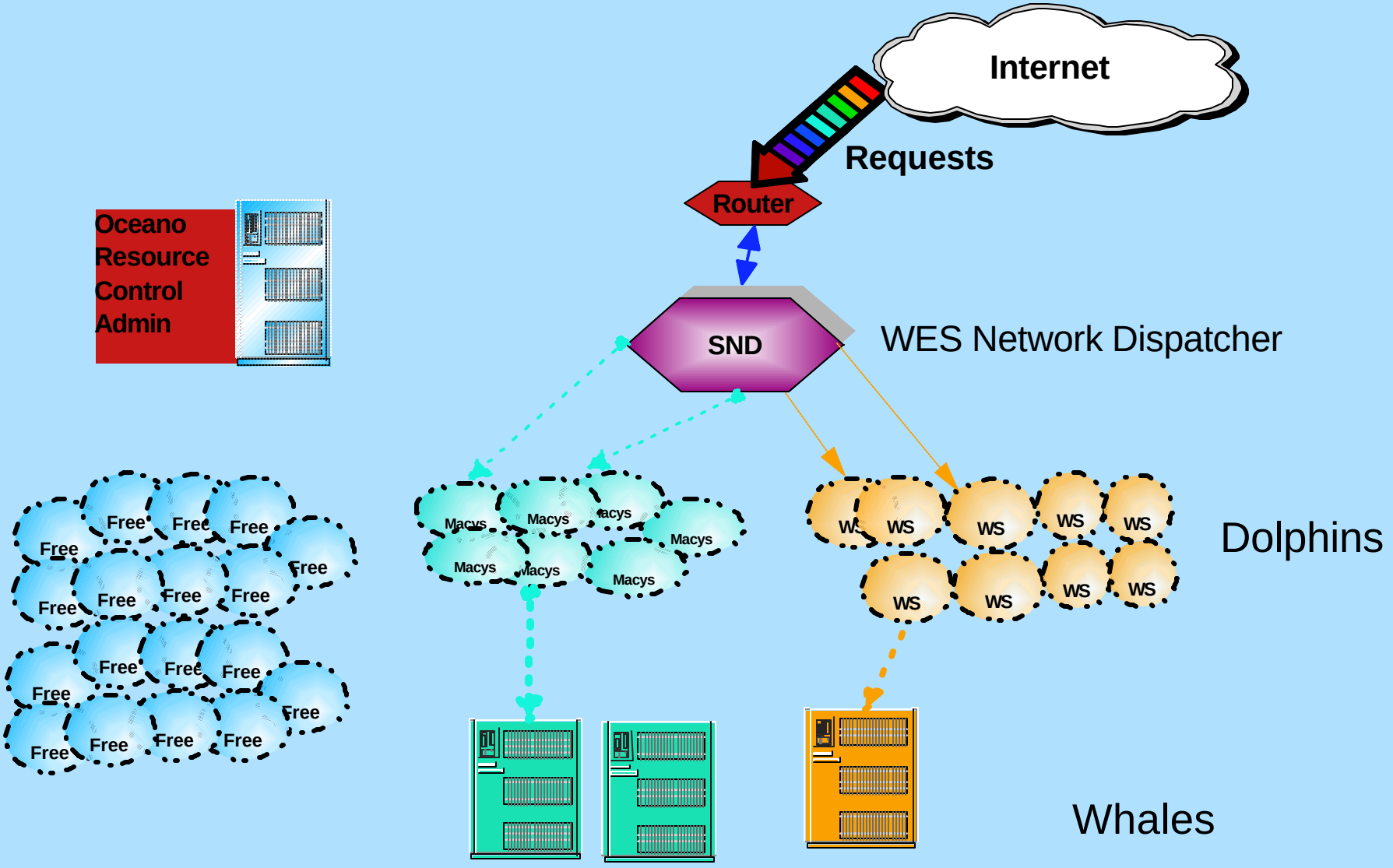
Characteristics

- Provisioning Platform
- Shared Infrastructure
  - Isolation for each realm
- peaks covered (autonomic)
  - rapid allocation of resources
- Automation
  - reduce administration cost

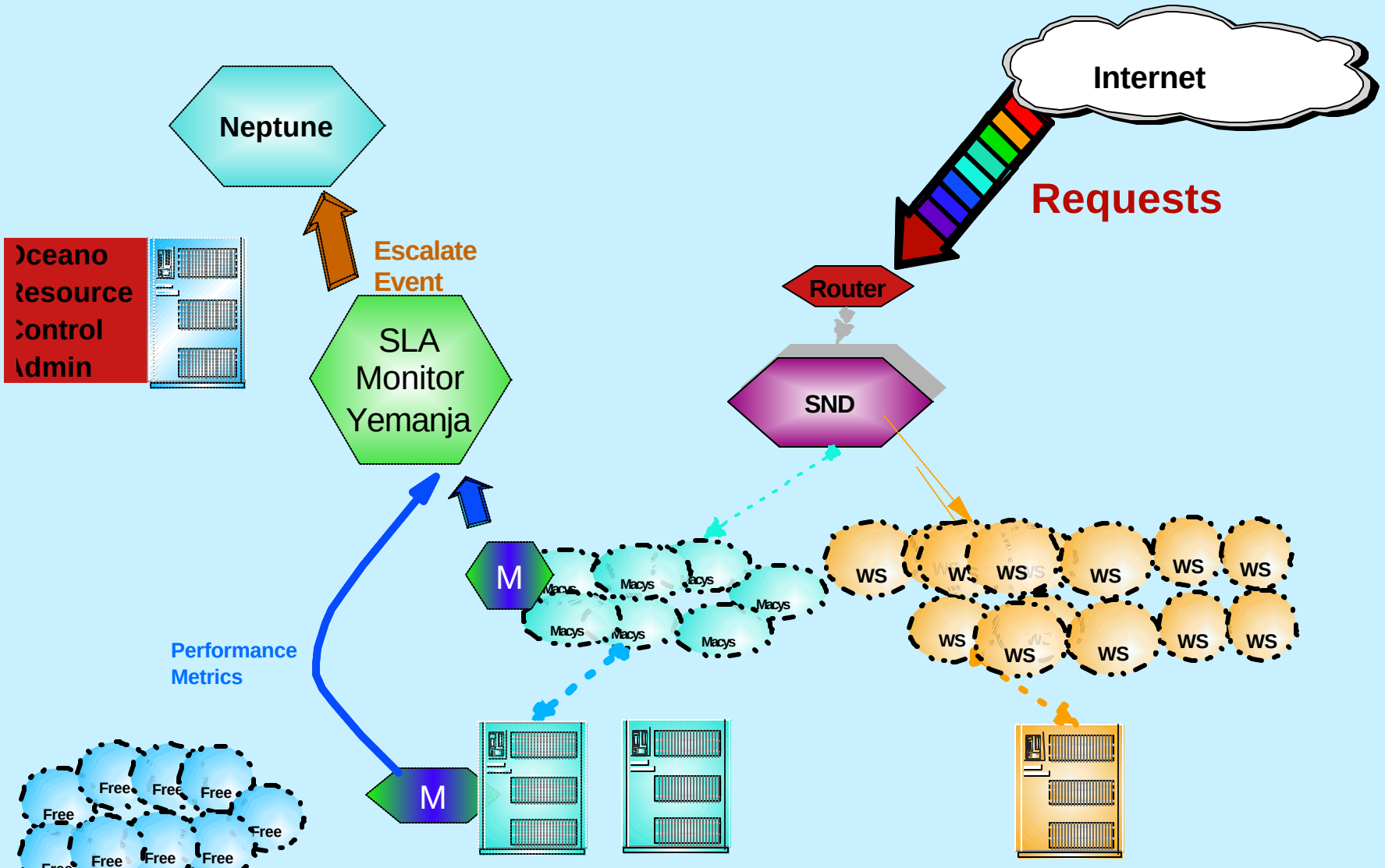
Océano Objectives

- Efficient infrastructure for eUtilities
  - Multi-customer hosting on a virtualized collection of resources
  - Drive down people management costs via automation
  - Scalable and highly available
- Handle spiky workloads ; provide capacity on demand
  - Automated, fast add/remove [clean, secure]
  - servers, bandwidth, storage
- Create Infrastructure SLA (ISLA) contracts
  - support dynamic resource allocation model
  - ISLA monitoring and enforcement
- Technology applies to several environments:
  - NetGen SPs, Enterprises, ...

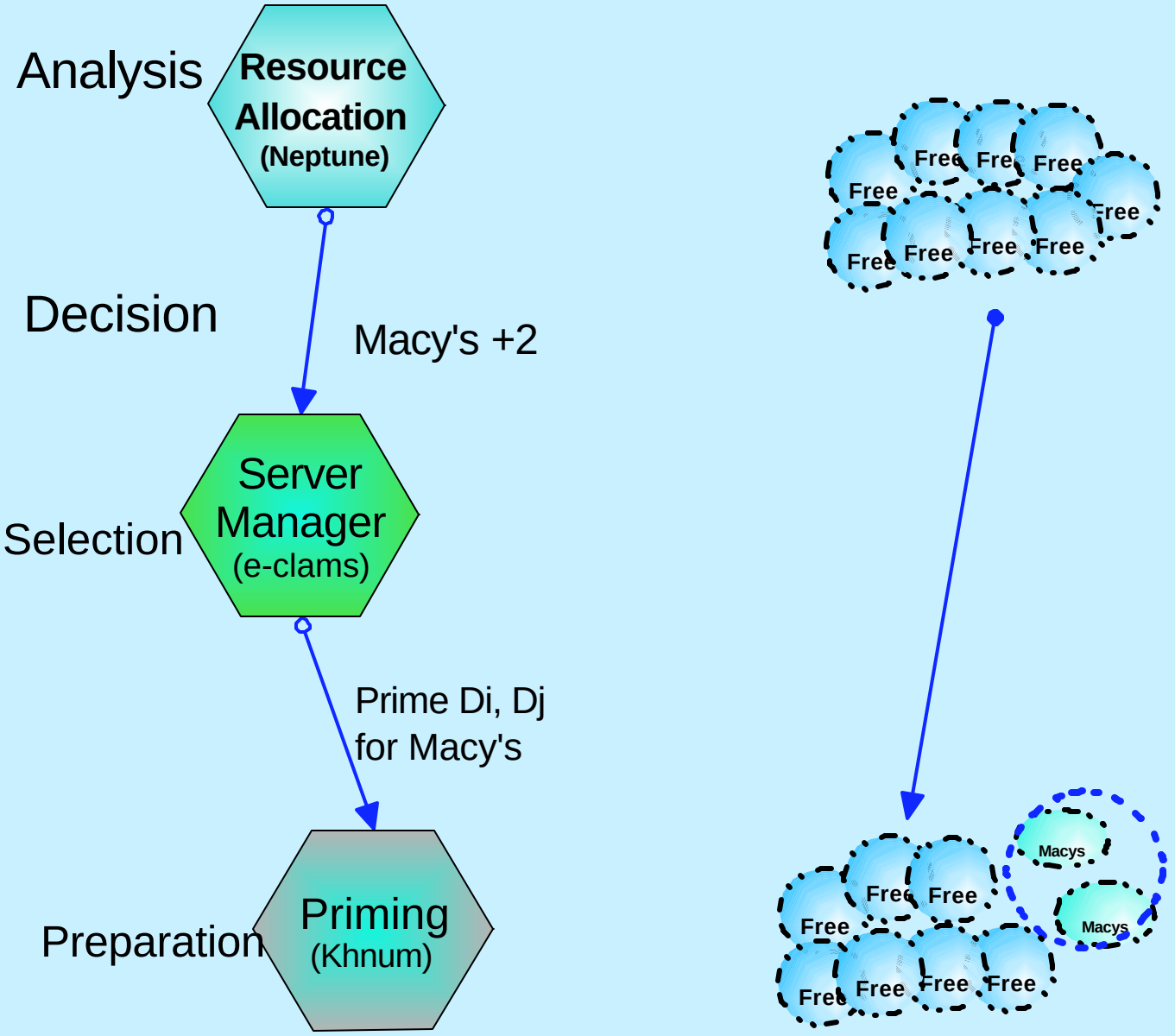
Flow of Requests into Server Farm



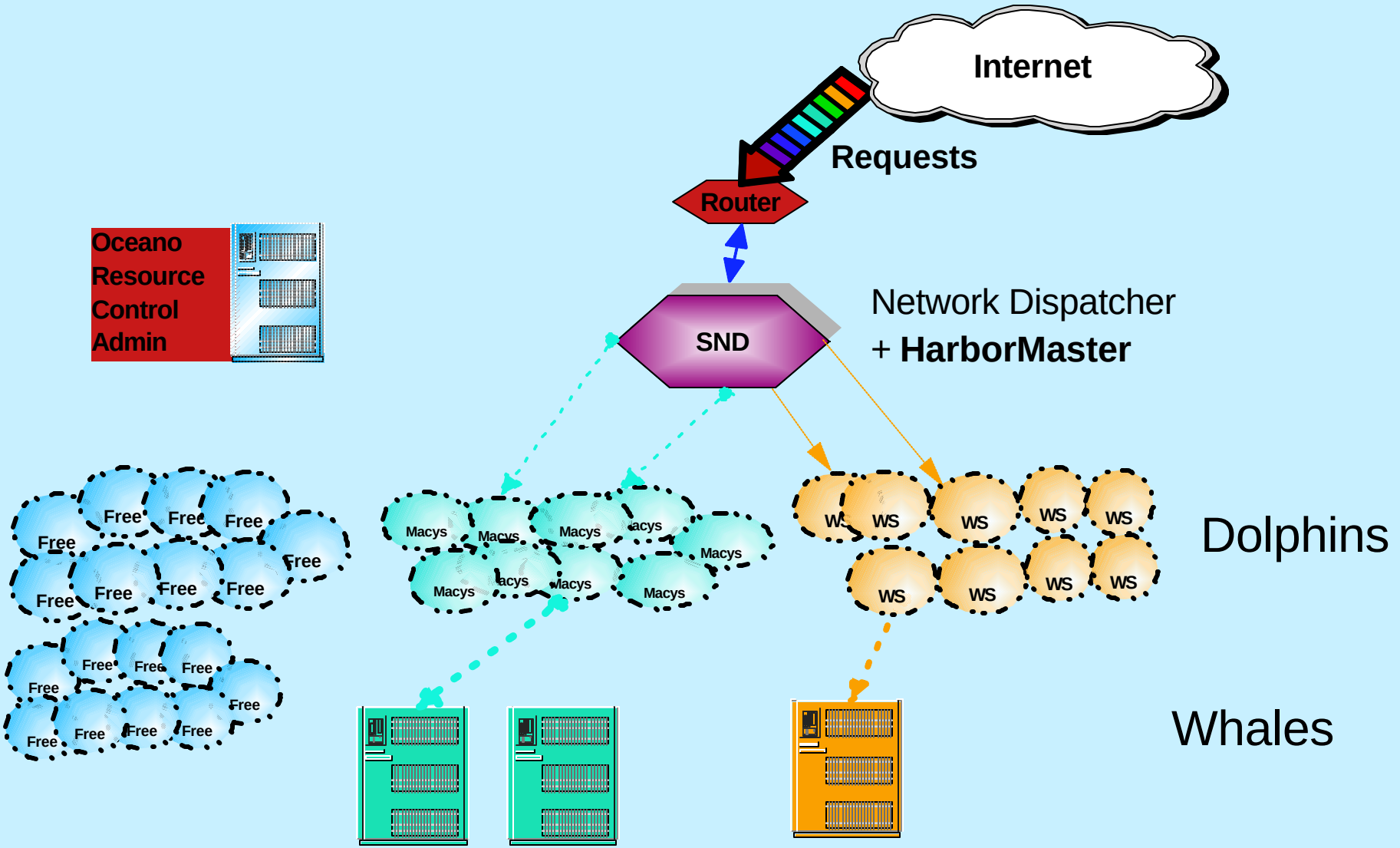
ISLA Monitoring



Oceano ISLA-based resource reallocation

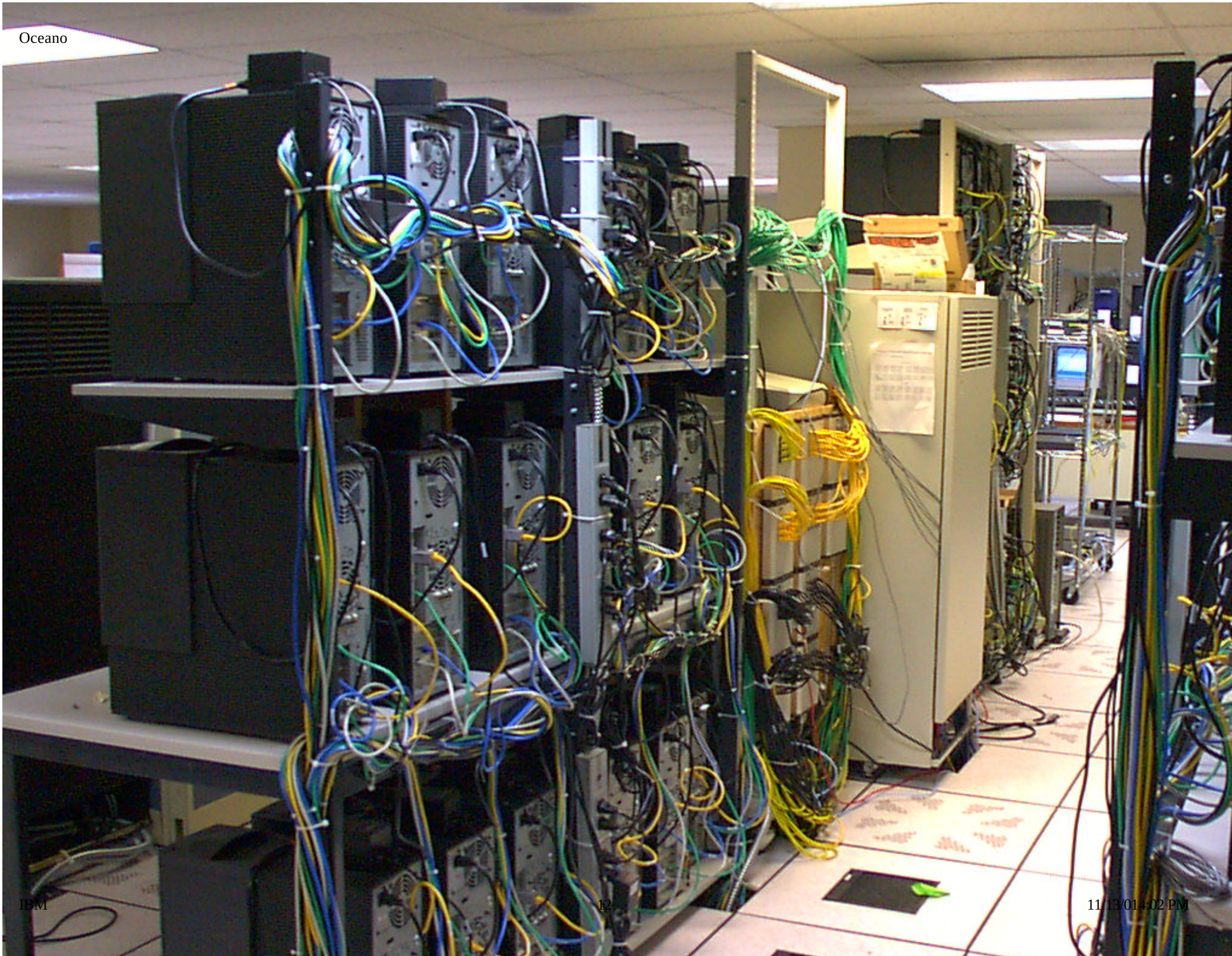


After the addition of 2 servers

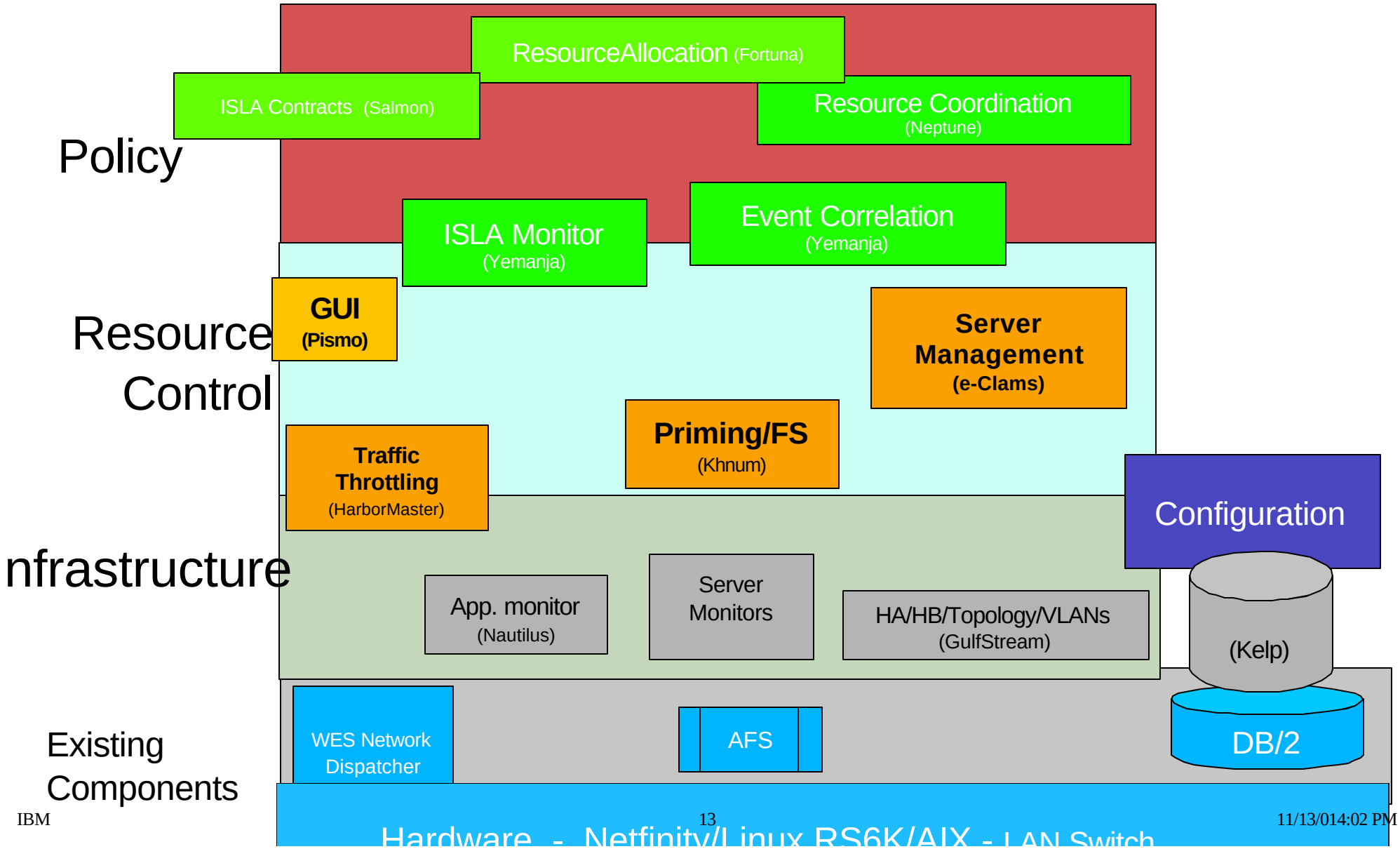



IBM
Océano


9



Océano Components

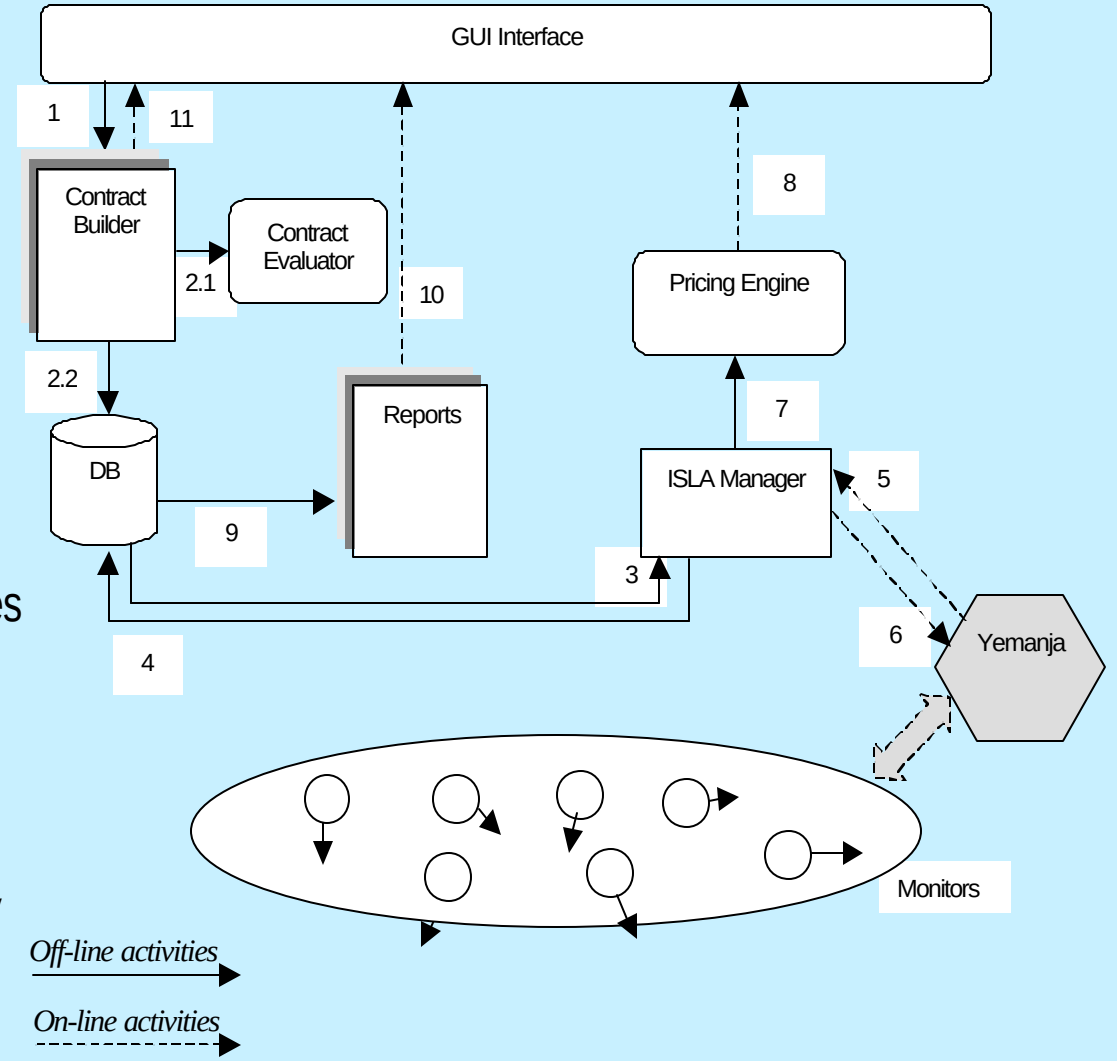


Policy Layer Components

- Salmon
  - Contract definition, pricing, billing
- Yemanja
  - ISLA monitor
  - Problem Determination (event correlation)
- Neptune
  - Resource coordination
- Fortuna
  - Intelligent Proactive Allocation

Salmon - Service Agreement Levels for Monitoring Océano coNtracts

- ISLA contract definition
- ISLA Manager
  - Response Automation
  - Violation Detection
    - Violator, Grace Period, Action/Penalty
- Pricing engine:
  - Flat-rate, Usage-based and penalties for violation
  - Standard Equations:
    - Charges: Contract Flat-rate, Usage-Based,
    - Sub Contract Addition, Penalty per Violation and prediction queries
  - Futures and Options



Yemanja - Event Correlation

- problem determination
  - hierarchical event correlation
    - hardware faults, application faults,
    - ISLA performance violations
- policy monitoring and violation detection
  - integrate detection with performance monitoring and problem determination
- automated violation handling
  - alert resource manager (Neptune)
  - open problem records

Yemanja - Problems to be addressed

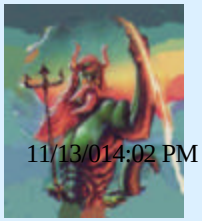
- Difficult to capture complex problem scenarios
  - Integrate HL ISLA violation with low level network monitoring
- Need method to propagate problems to all affected systems
  - recognize affected components
  - resist hard coding of dependency information
    - hard to anticipate all affected components
    - component models become large and unmanageable, adding new components can affect preexisting component models
  - cancel dependent problems when initial problem is fixed
- Simultaneous faults
- Uncertainty in causal implications
  - Lost and spurious alarms
  - Need for integrated testing
  - Scenario waits
- Dynamic system configuration changes

Yemanja Features

- ISLA violation detection integrated into correlation rules
- Rules can contain a mix of methods and events
  - Allows collection of additional data, or the analysis of state information before all required events arrive
- Associate and rank rules that represent alternate solutions to the same set of events
- Built in problem database
  - canceling root problem, cancels dependent problems automatically
- Flexible way to collapse multiple events of the same type to a single set based event specification
  - Can require that some % of resources in a resource-set generate the selected event

Neptune

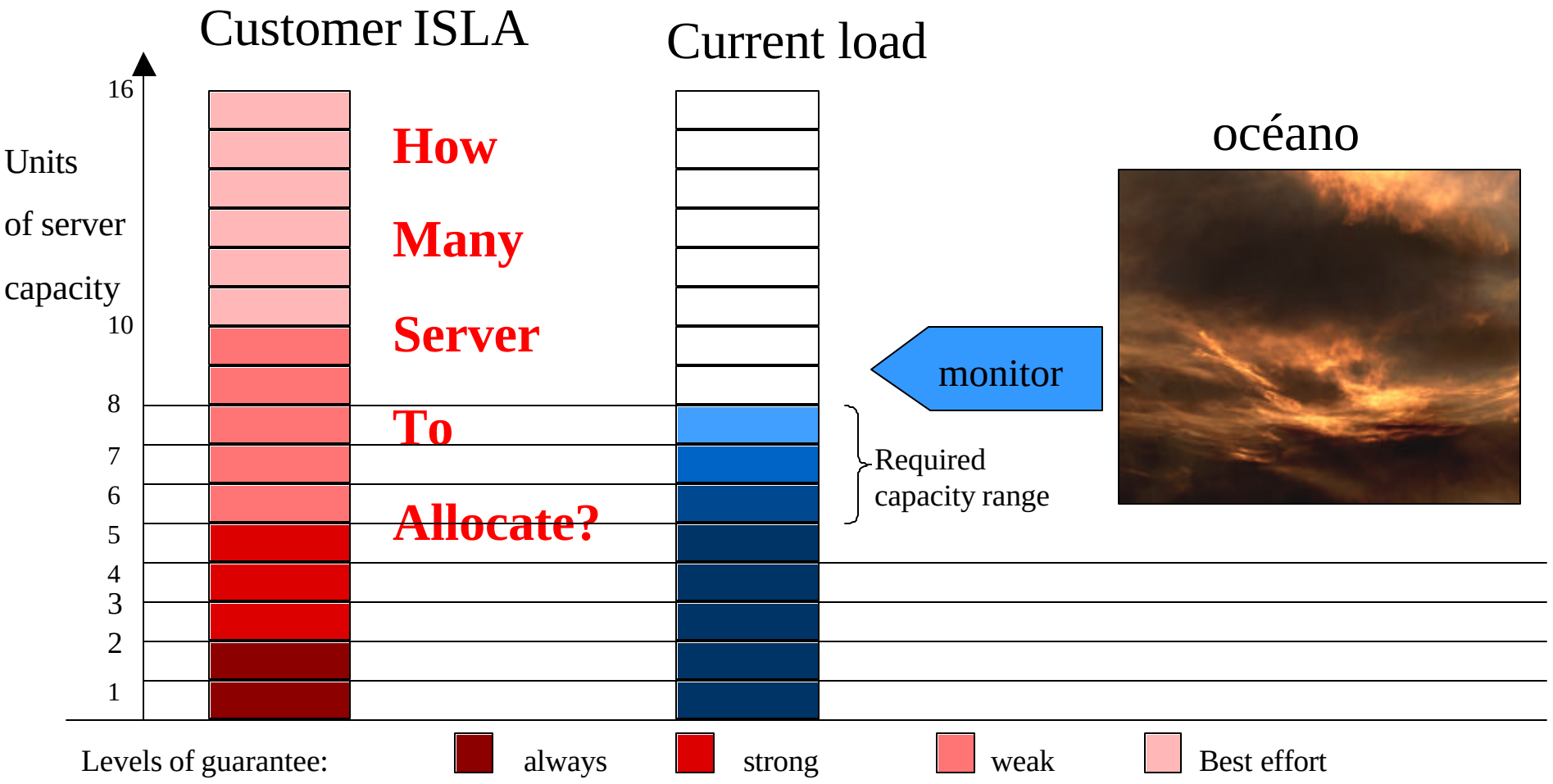
- Reactive resource allocation
  - plan based
  - allocates servers, bandwidth
- Reacts to
  - performance problems
  - component failures
  - ISLA violations
- Activated by Yemanja



Fortuna - Resource Allocation Strategy

- Goals:
  - Improve Performance + Maximize Revenue
- Planned + Reactive
  - Planned: use prediction of periodic traffic patterns
    - Construct a resource allocation plan (e.g. for the next 24 hours).
  - Reactive: (de)Allocation based on current load
    - Correct initial plan
    - give feedback to improve the prediction/analysis.
  - Operate in a fully reactive mode
    - for a new customer or
    - if the system observes unexpected behavior

Preliminary Example of a Layered ISLA



ISLAs and Revenues

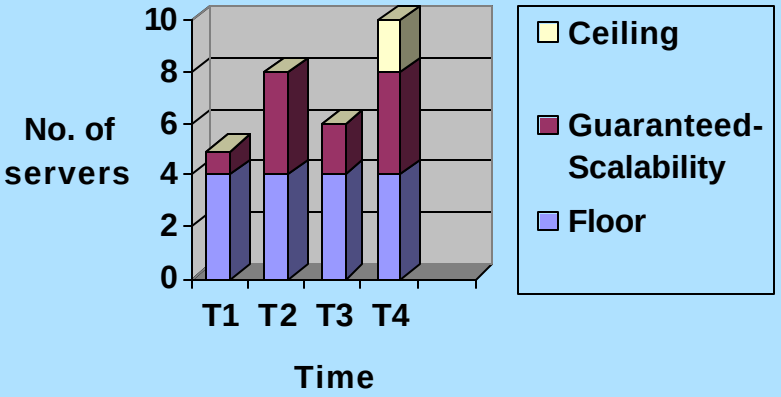
- Layered ISLA
  - Current state depends on the required server capacity, and state parameters (layer i):
    - maxi servers – the layer's boundary
    - Charge for capacity, time unit Ci
    - Penalty for a violation in this layer Pi size depends on the level of guarantee
- Options
  - Exercised implicitly according to measured load.
  - Price of an option depends on the level of guarantee, the capacity (maxi) and can also depend on the expected usage.

Scenario

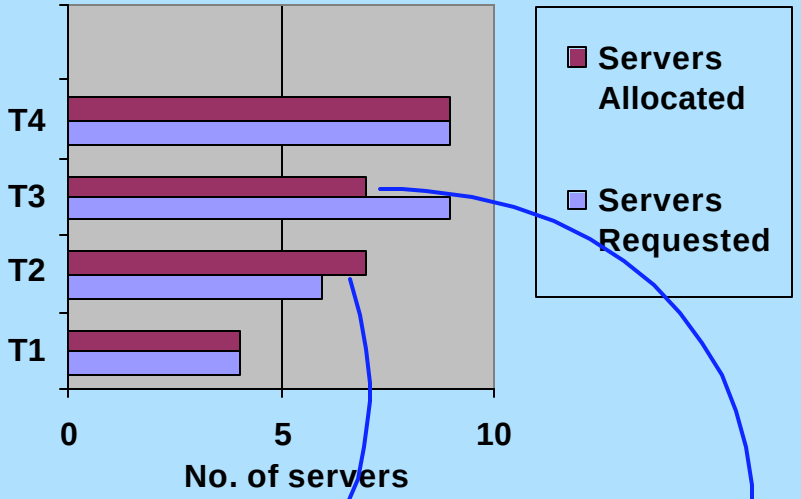
Active Scenario:
Scenario 1: {[Server_Set(4, 4, 2)], 00:00 Dec/01/2000, 23:59 Dec/31/2001, 1}

Definition Level

Allocation on the 3 levels of guarantee



Resources Requested X Resources Allocated



Monitoring Level

Over Provisioning

Violation

Usage-Based Charge Calculation

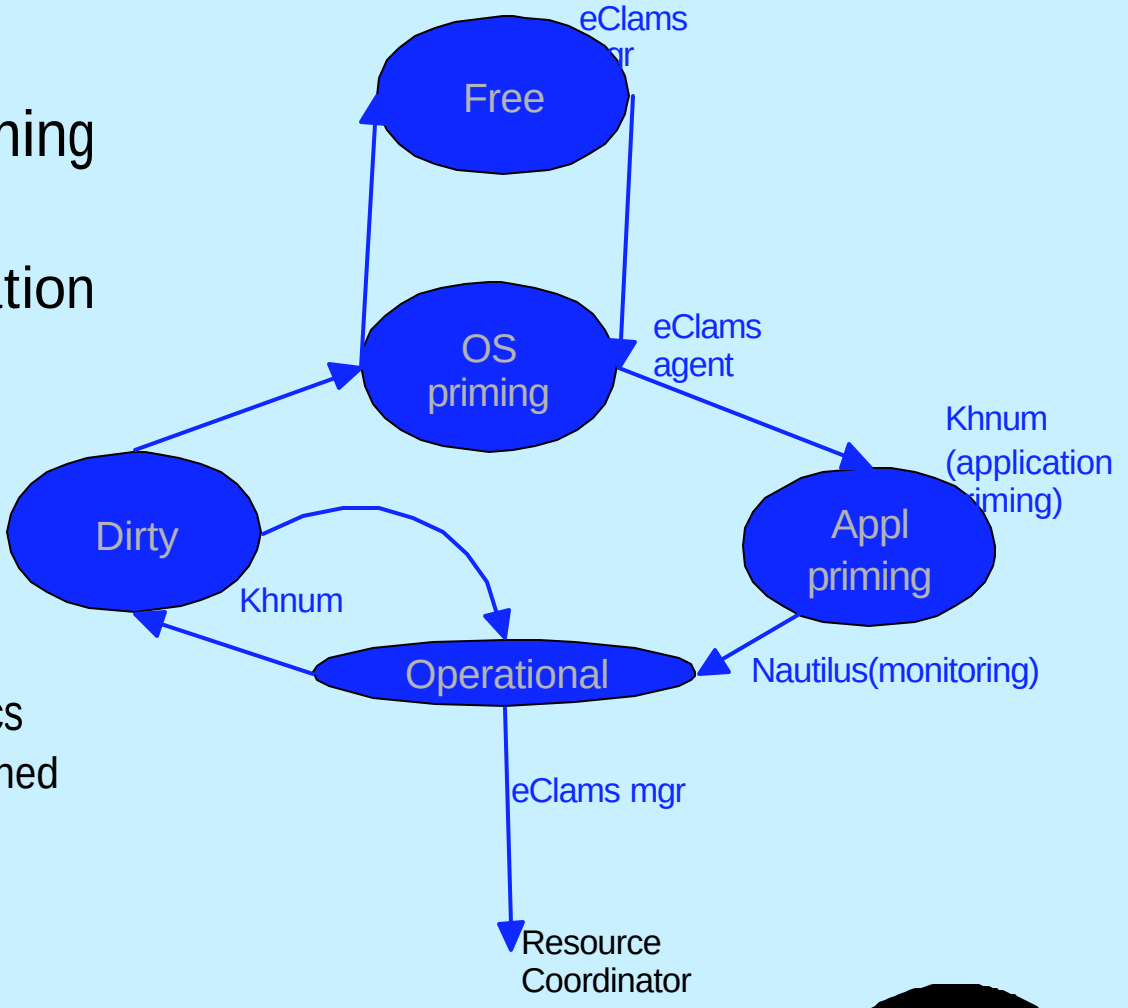
Charging Level

Resource Control Layer

- eClams
  - server allocation/reclamation
- Khnum
  - application and data priming
- HarborMaster
  - bandwidth management (request throttling)
- Pismo-Beach
  - GUI

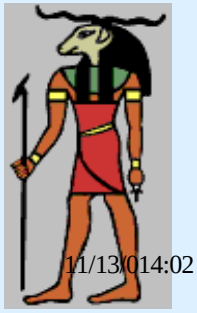
eClams - Server Pool Management

- Functions:
  - (De) Allocation and priming support for servers
  - Automatic network installation of OS (e.g. LUI)
- Future:
  - heterogenous server management
    - server specific characteristics
      - server capacity, server attached resources



Khnum: App + Data deployment

- shared file system
  - AFS-based (prototype)
  - few things are kept on local disk
- Cache Pre-loading
  - Removes load from File Server
  - "near local" performance
- Multicasting (MTFTP)
  - keeps network utilization low
  - multiple servers at once
- Cooperative caching
  - Accessing "neighbor's" memory faster than disk



HarborMaster - Workload Balancing

- No need for special hardware
  - Portable extension to Websphere Edge Server (aka Network Dispatcher);
- Load Balancing of Requests
  - Automatic Dynamic Reconfiguration of WES
- Throttle TCP connections
  - drop a percentage of requests
  - use of overflow server

GUI - Pismo Beach

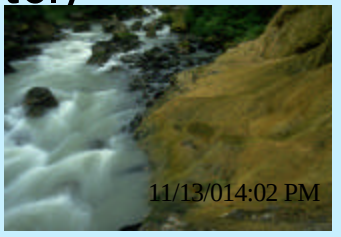
Displays: current status of servers, customer allocations, performance history, significant Océano events, component based tracing information

Infrastructure Layer Components

- GulfStream
  - Infrastructure monitoring
  - Network reconfiguration
- Nautilus
  - Application monitoring
- Kelp
  - configuration data management

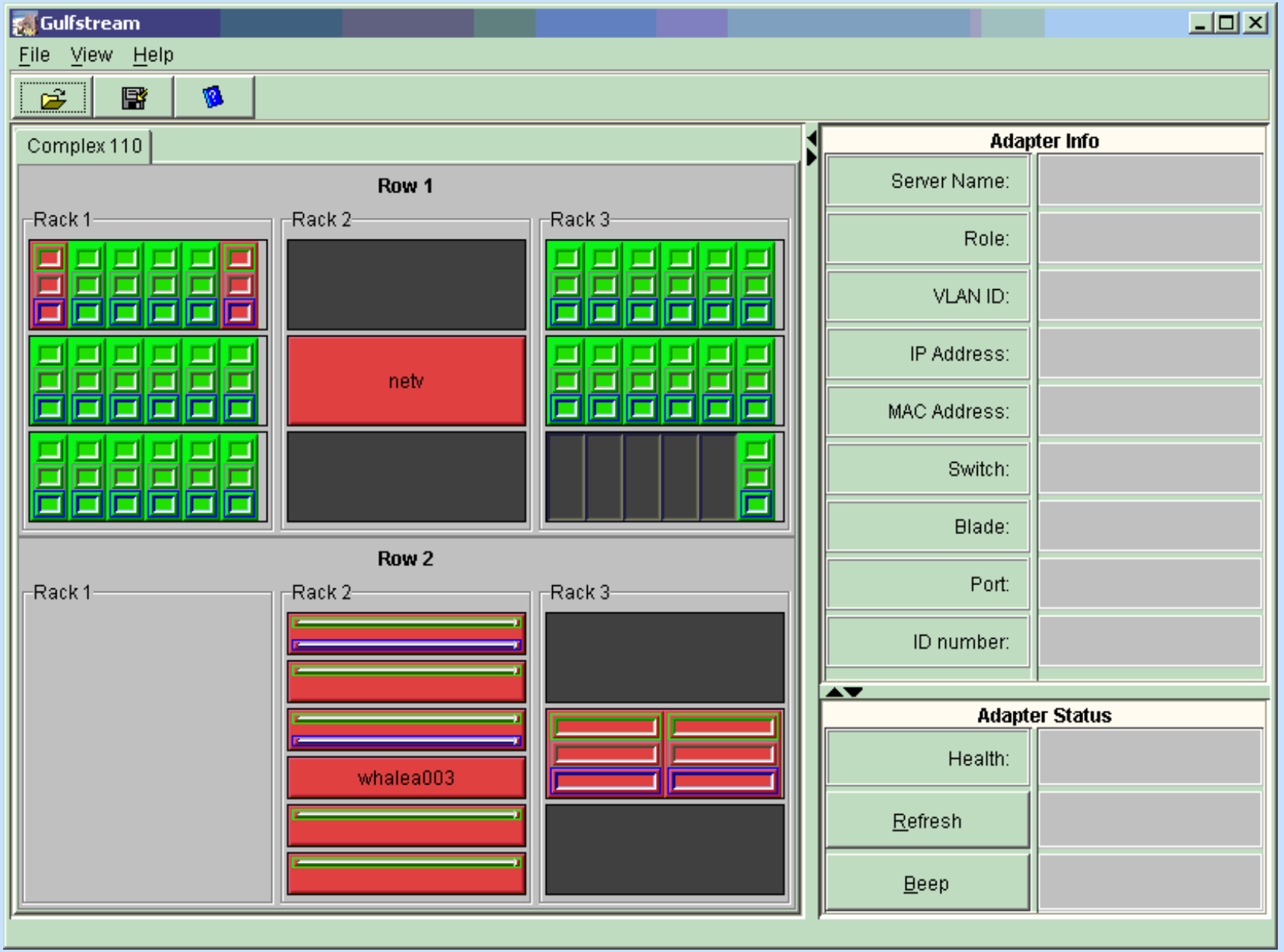
GulfSteam

- Function:
  - global topology discovery
    - via adapter discovery and correlation
  - failure detection via periodic heartbeats
    - Monitoring from *Within* the Server Farm, *every* NIC, *every* (V)LAN Segment
    - implied monitoring of *every* node
  - reporting up/down status of components
  - automatic reconfiguration of VLANs
- Future:
  - handling configuration changes
    - (new/changed hardware, network rewiring, maintenance, etc.)
  - Scalability to 1000's of nodes (BlueStream)



Diagnosis

Displays: status of servers & adaptors, position in physical racks



Nautilus

- Functions:
  - Application Monitoring
    - Traffic class specific monitoring
    - Response time, data output and request rate
    - Generates threshold based alerts
    - Node metrics correlated with traffic monitoring
  - Content Based Throttling
    - Admission control of requests based on request content

Kelp

- Functions:
  - configuration database + access methods
  - Dynamic reconfiguration and ISLA data
  - Object based interface to data
    - local cache of data and interrelationships
- Future:
  - automated cache updates (trigger driven)
  - additional verification

Prototype Status (Oct 2001)

- Lab:
  - 81 servers: 75 Intel/Linux, 6 RS6K/AIX, CISCO CAT6509
- Linux/AIX prototype of Océano supporting:
  - Service level monitoring (simple ISLAs)
  - Autonomic allocation of Linux servers
  - Automatic, scalable priming
  - Bandwidth management
  - Automatic discovery of network connectivity
  - VLAN management
  - Appl. monitoring and content based throttling (Apache)
  - Display of status, history, performance, events