Agenda
Turbonomic Introduction
Architectural Lessons Learned
Organizational Lessons Learned
Q&A
Turbonomic Introduction
A Marketplace of Buyers and Sellers

- vApp
- Provider VDC
- Org VDC
- Physical Machine
- Switch
- Network
- Data Center
- VM
- Container
- Storage
- Array
- vPOD
- dPOD
- Zone
- Region

Copyright 2016, Turbonomic
A Marketplace of Buyers and Sellers

Service Entity

Buy

Commodity

Sell

Copyright 2016, Turbonomic
Unified Demand-Driven Control Platform
Proof Points in The Journey…
Turbonomic Legacy Platform

- Database
- JVM
  - UI Services
  - Analytics
  - Abstraction
    - Repository
  - Mediation Components
    - VC
    - UCS

- Monolith JTomcat
- Eclipse Modeling Framework
- Home Grown Repository
- MedCom

Copyright 2016, Turbonomic
MonolithFirst

Going directly to a microservices architecture is risky

A monolith allows you to explore both the complexity of a system and its component boundaries

As complexity rises start breaking out some microservices

Continue breaking out services as your knowledge of boundaries and service management increases

Copyright Martin Fowler
Summary of Initial State

- Monolithic Architecture
- Monolithic Team Structure
- Release Every Six Months
- Few Metrics Captured
Catalysts for Change

Growth in Customer Base – to over 1500
More Large Environments – up to 100K VMs under management
Geographical Spread of Team – US / Canada / Italy / Greece / Russia
More Frequent Deliveries – Several Times a Year vs. Every 6 mos
Expanding Feature Base – Across the Datacenter
Current Turbonomic Challenges

Turbonomic Today
• In-Memory Repository
• First-Generation Analytics Algorithm
• Monolithic Services Platform
• Flex UI Technology

Consequences
• Real-Time Capacity Management & Planning at Scale
• Requires Multiple Turbo Instances Plus Aggregator
• System Management / Upgrade Complexity
• Performance Bottlenecks
• Security Issues
Lesson 1: Adopt and Apply Architectural Principles
**Principle**

Separation of Concerns:
A component should only implement the business function it is mandated to cover and not be concerned with other issues. For example, an Analysis component contains code to compute a set of actions.

**Benefit**

Components can easily and quickly be modified, or even removed and replaced. This could be for a number of reasons, e.g.

- **Upgrade** – as functionality is extended and efficiency increased, the user can take advantage of the latest version
- **New technology** – as technology changes, it can be quickly incorporated into new modules, completely replacing the old
Principle

Façades
*The primary requirement is that all components are known through their interfaces.*

Benefit

Clear Interfaces – Independent Implementation; More Robust Testing, less confusion for developers
Principles

**Runtime Dependency Resolution**
If component A depends on component B (e.g., Market depends on a Pricing Function), B is provided to A rather than A being coded with the way to find B. This results in concise and configurable components.

**Code Instrumentation**
Components will need to be instrumented in order to be managed at runtime. Java components should be manageable using an API that they expose.

**Versioning**
Services and API Endpoints can be versioned in configuration so that upgrades to software can be managed and controlled.

Copyright 2016, Turbonomic
Benefits

- **Maintenance**: Upgrade, re-configure and examine Individual Components through a single configuration pane-
- **No Need to Stop Everything**
- **Delivery**: Each component Runs in A Container and can be independently updated
Well-formed Components
Well-formed components are those that adhere to the principles listed above. In addition, well-formed components are delivered with built-in, non-regression functional and performance tests. In addition, standard documentation is packaged with components and may be browsed once deployed.

Additional Benefits
- **Scalability**: new Algorithms, and technology choices, Horizontal Scale-Out for Increased Capacity – ability to visualize and control arbitrarily large datacenter estates in a single instance

- **Performance**: Each component runs in its own container and Turbonomic will find the best place for it.

- **Resilience**: Architecture enables HA

Copyright 2016, Turbonomic
Lesson 2: Borrow Freely
What is a Turbonomic Component?

Carries out Specific a Function
  (e.g., Mediation, REST API, Analysis)
In addition it Implements interfaces for
  • Manageability
  • Messaging
  • Resilience
  • And it runs in a container
Key Interfaces: Resilience

- VMTComponent—Resilience Support

- Purpose: Component Life-Cycle Management

- Provides
  - Start
  - Stop
  - Restart
  - dumpData
Key Interfaces: Management

• VMTComponent – Manageability Support
• Purpose: Manage components ‘in flight’
• Provides
  • Status Assessment
  • Runtime Configuration Changes
Control Plane Architecture

Control Plane Library
- REST API to Outside World
- Lifecycle
- Registration
- Security / Authentication
- Configuration Changes
- WebSocket Endpoints –
- Maintenance UI API
- Logging / Flow Tracing / Auditing
- Statistics

Components
- Called by CPL - Lifecycle/Config
- Calls CPL - Registration/Websocket
- Packaged with CPL as Docker Container

Shared, Persistent Configuration
- Properties For Each Component
- Current Status For Each Component

Copyright 2016, Turbonomic
Leverage Modern Open Source Infrastructure

"Buy" Rather Than Build;
Many Ways to Get Networked Architecture Wrong

Service Architecture Functions:
• Service Registration / Discovery
• Networked Configuration
• Intelligent Routing
• Circuit Breaker
• Client-Side Load Balancing
• Distributed Tracing
Control Plane Architecture Enablement

**Spring Cloud**
- REST API to Outside World
- Lifecycle
- Registration
- Security / Authentication
- Configuration Changes
- WebSocket Endpoints –
- Maintenance UI API
- Logging / Flow Tracing / Auditing
- Statistics
Control Plane Architecture Enablement

Consul

• **Service Discovery**: Clients of Consul can provide a service, other clients can use Consul to discover providers of a given service.

• **Health Checking**: Consul clients can provide any number of health checks

• **Key/Value Store**: Multipurpose Key Value Store
Lesson 3: Document
Even the Basic Stuff

Copyright 2016, Turbonomic
Example: Creating a New Component
docker container

Topology Processor Component
Component Base Layer
Jetty

docker container

Mediation Component
Component Base Layer
Jetty

docker container

Market Component
Component Base Layer
Jetty
Lesson 4:
Don’t Abandon the Monolith Too Quickly
Technical Debt
Define initial set of Clean interfaces In the monolith itself

Define Reusable Components
Dual Use Components

Legacy Monolith
- UI (Flash)
- M1 1.0
- Legacy Mediation
- Obsolete Frameworks (EMF)

New UI
- M2 2.0
- VC SDK
- Hyper-V MedCom
- MedCom Prebe
- OpenStack Prebe
- REST API v2

XL
- Distributed Components
- Scalable Repository
- Generalized Scale out
- HA

Performance
- Single instance scaling limits
- Stability
- Ops Mgr Management
- Aggregator functionality gaps

>1M Entities
- Better Manage ability
- Manage Large data centers

Copyright 2016, Turbonomic
Lesson 5: Don’t forget about Security
Security Concerns

• **Open Source** - *Determine and Correct Vulnerabilities:*

• **Container Hardening** – *Containers should have a minimal set of services. Known Vulnerabilities should be eliminated.*

• **Inter-service communication** - *Secure it*
Microservice authentication, and authorization

- **AUTHENTICATION** microservice
  - Runs isolated in its own container
  - Validates user credentials and in the future microservice credentials
  - JSON Web Token (Stateless)
  - Similar to OAUTH2 which doesn’t specify a token format.
  - Decentralized, non-persistent authentication and authorization
  - JWT contains user authorizations/roles in the payload
  - JWT is signed by Auth Service and can be verified authentic
  - Payload does not contain any sensitive information.
  - JWTs passed among services (similar to Kerberos)
Microservice authentication, and authorization

- Stateless
  - Token contains all the information required to identify the user, and permissions can be added to the payload.
  - No need to maintain session state
  - User can be passed to any microservice copy. Does not need to be bound.
- Reusable
- Secure
  - Cookies not required (CRSF attacks do not exist)
  - Can encrypt tokens (JWE) if any sensitive information is carried
  - Transmit over HTTPS to prevent MIM attacks
- Fast / Efficient
  - No server side lookups required to deserialize session on each request.
  - Calculate the HMAC SHA-256 to validate the token and parse content

Copyright 2016, Turbonomic
Lesson Last: Test, Build, Test, Measure and Release Often

Copyright 2016, Turbonomic
Organizational Principles for Microservices

Peer Review

Code Quality Measurement:
WTFs/Minute

Good Code

Bad Code

http://commadot.com

Copyright 2016, Turbonomic
Agile

- Daily Scrum Meeting
- 24 hours
- 2-4 Weeks
- Potentially Shippable Product Increment
Mediation Component: “MedCom”

Functionality:
Retrieve data from customer env

Key features:
- Target type specific
- Implemented using SDK framework
- Using mediation API

Copyright 2016, Turbonomic
Functionality:
- Orchestrate Mediation Component data retrieval
- Merge/validate data coming from MedComs
- Edit topology using policies/settings
- Present consistent topology to repo + m2
Group/Settings/Policy Manager

Functionality:
- Group/settings/policy bookkeeping …
- … providing them as needed

Copyright 2016, Turbonomic
Deployment Manager

**Functionality:**
- Reservation orchestrator
- Deployment orchestrator

---

Turbonomic.com

Copyright 2016, Turbonomic
Repository

**Functionality:**
- Store/retrieve topologies
- Searching capabilities

**Key features:**
- Uses graph database (ArangoDB)

---

*MedCom Copyright 2016, Turbonomic*
Functionality:
- Run analysis on a topology
- Generate actions toward desired state

Key features:
- Fast
- Flexible price function definition
- Agnostic to entity and commodity type
Action Orchestrator

Functionality:
- Store/retrieve actions
- Orchestrate action execution

UI
- Plan Orchestrator
- Repository
- Deployment Manager
- Market
- Topology Processor
- Group/Settings/Policy Manager

MedCom

Copyright 2016, Turbonomic
Plan Orchestrator

Functionality:
- Store/retrieve plans
- Orchestrate plan execution

UI
Action Orchestrator
Market
Repository
Deployment Manager
Group/Settings/Policy Manager
Topography Processor
MedCom

Copyright 2016, Turbonomic
**Functionality:**
- Show environment data ...
- … and actions for desired state

**Key features:**
- Html 5 + Javascript
- AngularJS framework
- Uses REST API
Fin