Abstracting Dynamic Routing with Workflows

Nigel Warren - Brunel University & Zink Digital
Ewan Silver – Autonomic Systems Ltd

Nige@zink-digital.com
ewan@autonomicsystems.co.uk

Background

- Spray - JCM10
  - Distributed Rendering
- Pipelining and Object Flow - JCM9
  - Virtual processing pipelines
  - Cells and Shapes
- Master Worker Patterns
- Simple Distribution
- “Many Things” - Fault Tolerance
  - Flocking and Swarming
  - Evolution and Genetics

Motivation

Some patterns of use of Jini and JavaSpaces can be assumed into a platform.

At the university the

MultiMedia Processing

BioInformatics

teams are neutral about distributed infrastructures …

… but they do care about processing vast amounts of complex information quickly.

Motivation

An interface that is ‘insanely’ simple to comprehend & use

To move platform innovations to …

Deploy,
Dynamic Routing,
Fault tolerance,
Evolution,
other things …

Motivation

Just don’t want to write ...

```
txn = TxnMgr.getTxn()
space.take(template, txn )
do work ( or work.do() )
space.write(work, txn )
txn.commit()
```

... even in pseudo code.

Applies to Pipelines too.

Applicability

Distributed Worker (aka Master/Worker)

- Rendering (Spray)
- Monte Carlo Simulation (VAR)
- Protein Folding

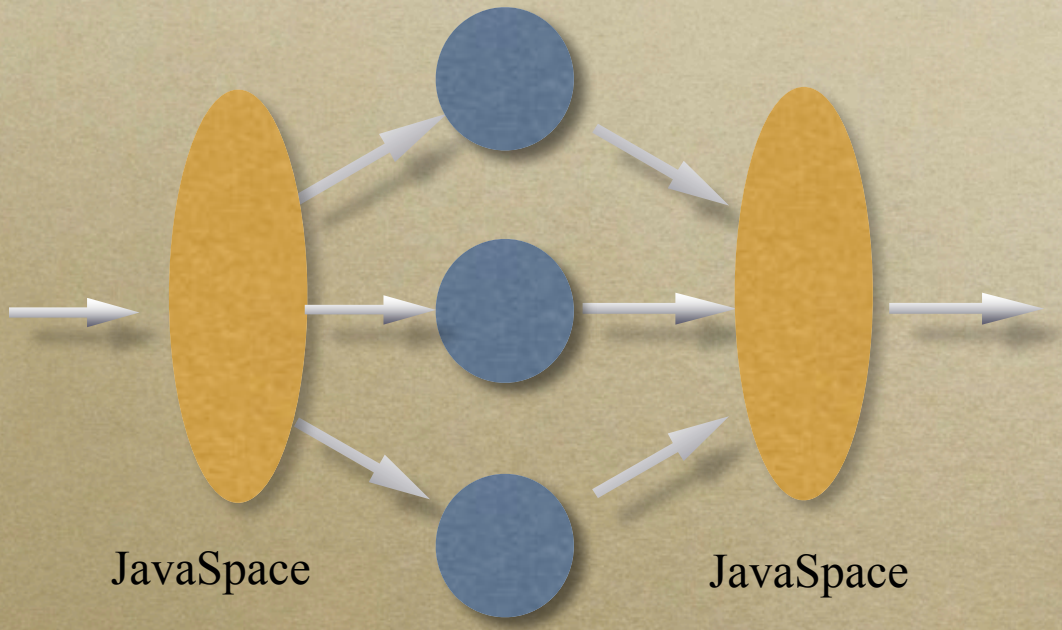
Pipelines

- Image (signal) processing
- Geometric transformations
- Business “Workflows”

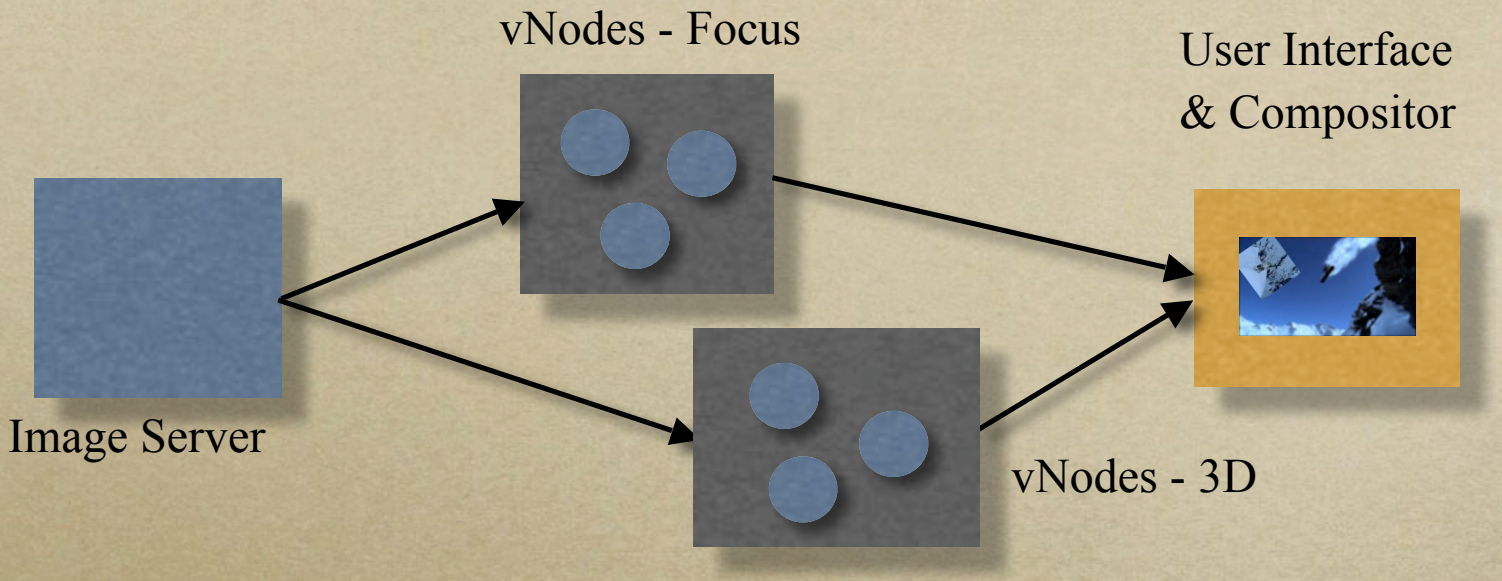
Distributed Worker patterns is a complete subset of the Pipeline pattern.

Distributed Worker as a Buffered Pipeline

Nodes are ‘Java code’ workers and buffers are JavaSpaces



Pipelines


vNodes - Focus
User Interface
& Compositor
Image Server
vNodes - 3D

Clifton Interface - Activities

Application writers implement ‘Activities’ (like ‘main()’)

```
package com.exocute.clifton;

import java.io.Serializable;

public interface Activity
{
    Serializable process(Serializable input,
                         ActivityToolKit atk);
}
```

Clifton Interface - Ambits

Application writers use Ambits to seed and gather objects.

```
public interface Inward  {
       public void put(Serializable obj);
}

public interface Outward  {
       public Serializable get();
}
```

In Clifton all object flow is buffered at each pipeline stage.

Clifton Implementation

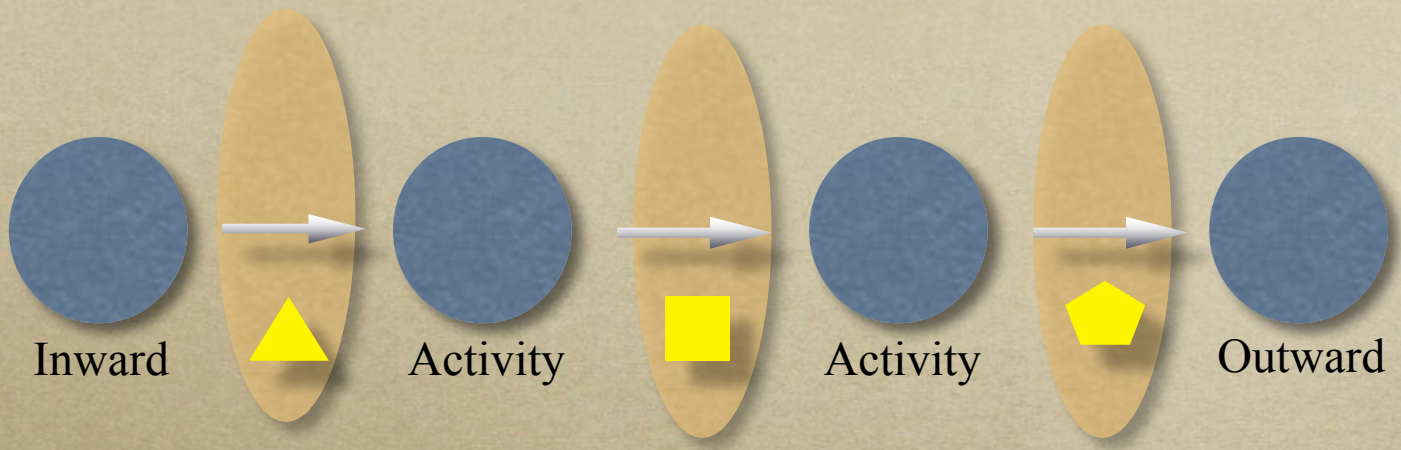
Virtual Nodes (vNodes) are jini services and execute `Activities` on host hardware.

Buffering, pipelining, distribution is handled by simple channel model and Blitz JavaSpaces

- ‘Lossy’
- Relatively low complexity
- Only Java based routing

Interface could be implemented without Jini ?!

Connections



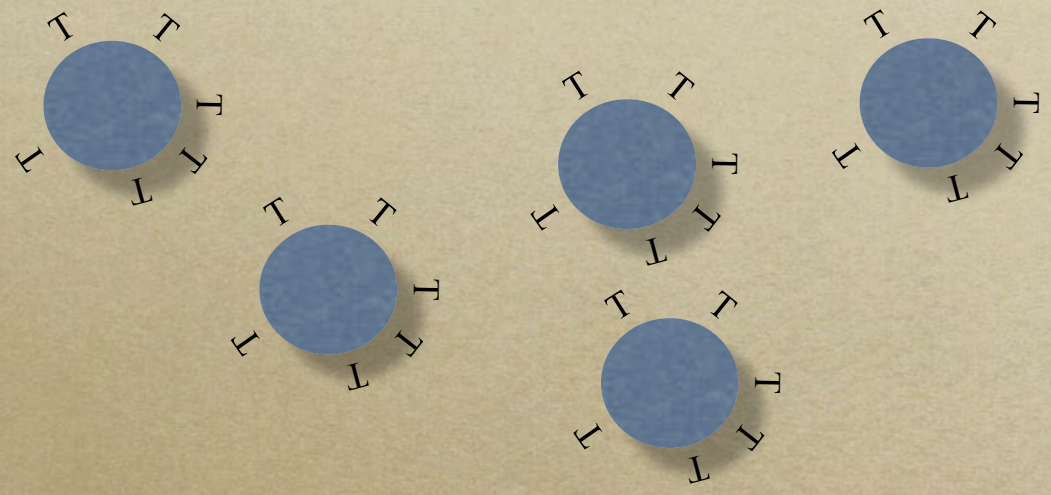
Routing Options :

Java Code

Routing Language

GUI - Local simulator

Cells and Shapes



Biological cells use ‘shapes’ on the outside of their nuclei to identify and interact with one-another.

Clifton uses shapes for routing

Clifton Stack

Routing Language
GUI
Local Simulator

Media Kit

Spray

L-systems

Clifton
Interface

ActivityToolkit
vNode
Paths

Channels
Buffers
Nodes

Jini - Blitz JavaSpace

Clifton View

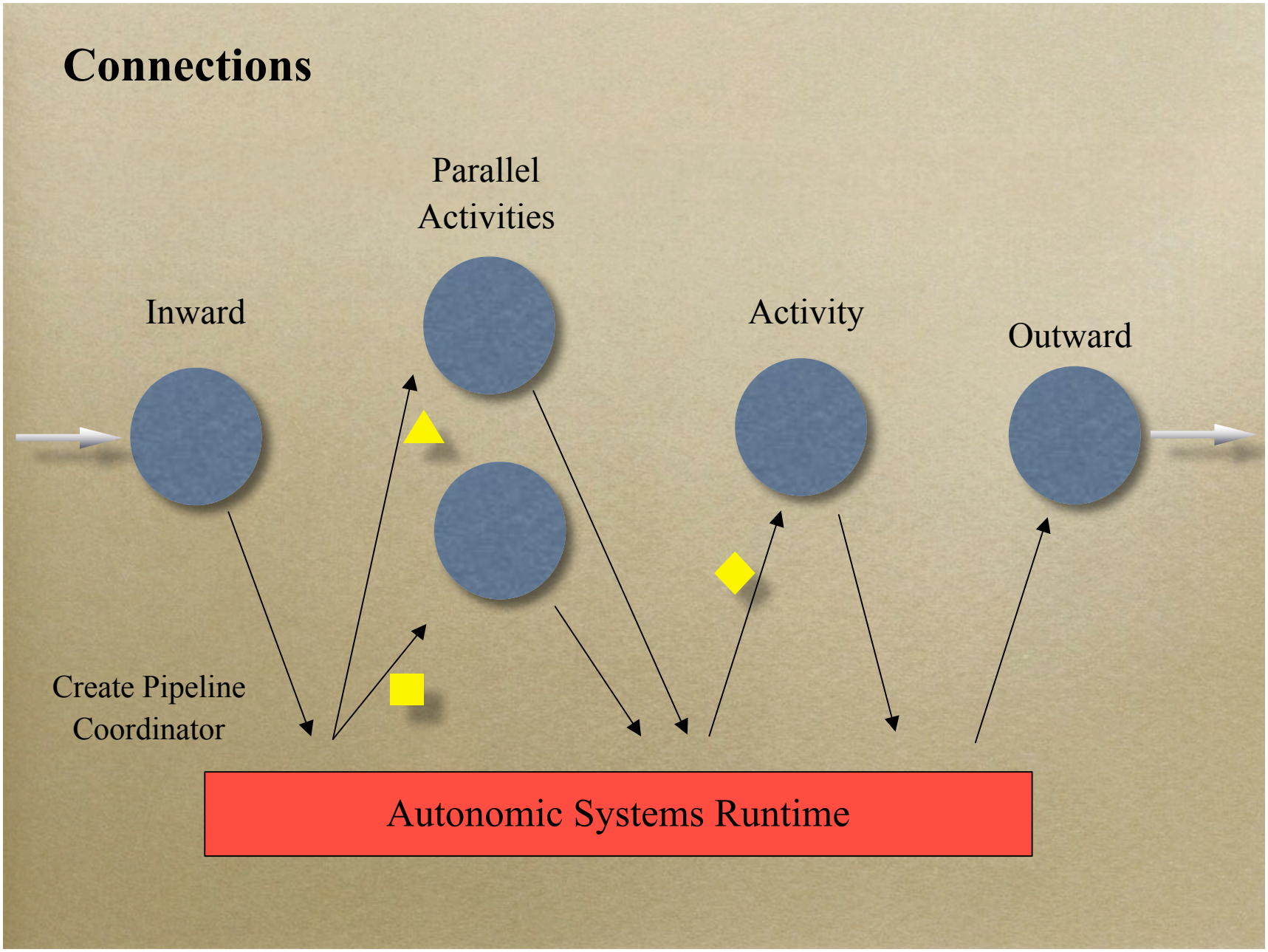
Autonomic Systems Stack

- Aimed at longer term pipelines – business workflows
- Pipelines could last for weeks, months or years
- Speed not so critical
- Reliability of data flow
- Activities can often run in parallel
- Changing business conditions require ability to dynamically evolve pipelines whilst data in transit

Autonomic Systems - implementation

- PipelineCoordinator :
    - configures and orchestrates Activities
    - dynamically evolve pipeline
- Inward ambit creates unique pipeline per invocation
- Activities routed and invoked via “Shapes”
- Transactional under the hood
- Transactional ToolKit - “over the hood”?
- Runs on Jini, Blitz (persistent) and Mahalo

Connections



Proof of Concept

Green field

- L-System compiler to simulate Algae growth
- Pipelines and Distributed Worker

Spray Port

- 1 hour 25 Minutes
- 40% reduction in complexity
  - Class count - LOC count

Genetic Algorithms

- Loops in the pipeline

Outlook

Aiming to make local simulator (threaded) available

- debugging class casts
- pipeline hinting

Routing Language - MediaToolKit

More applications from various domains

New platform implementations/upgrades

- Instrumentation
- Reliability

Thanks for Listening