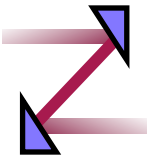
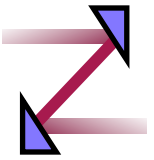


# Utility Computing with Amazon EC2



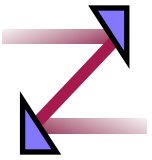
# Obligatory Intro?

- Distributed Systems Consultant
  - eTapestry
  - Vertex Pharmaceuticals
  - BT
  - Orbitz
- Apache Jini Committer
- Winner Jini Community Contributors Award 2004
- Author of Blitz JavaSpace

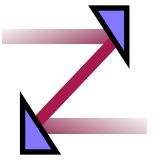


# Agenda

- Utility vs Other Models
- EC2 Overview
- Challenges working with EC2
- What I've done so far
- What I'll look at next
- Wrap-Up

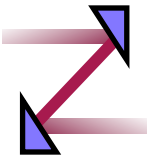


# Why's and Wherefore's



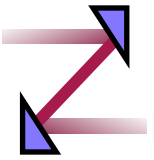
# Conventional Hosting

- “Permanent Residence”
- Static setup
  - Hardware changes rarely
  - Deployed software doesn't move
  - Fixed network infrastructure (IP addresses etc)



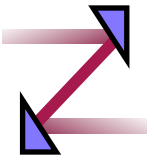
# Utility

- “Temporary Residence”
- Dynamic (unpredictable) setup
  - Hardware changes whenever
  - Deployed software can move
  - Dynamic network infrastructure
- No “special” machines



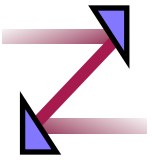
# Why the Differences?

- Utility is structured to support micro-billing
- Utility is “on-demand”
- Utility is lightweight
  - Minimal staff requirements from “host” (e.g. Amazon) perspective

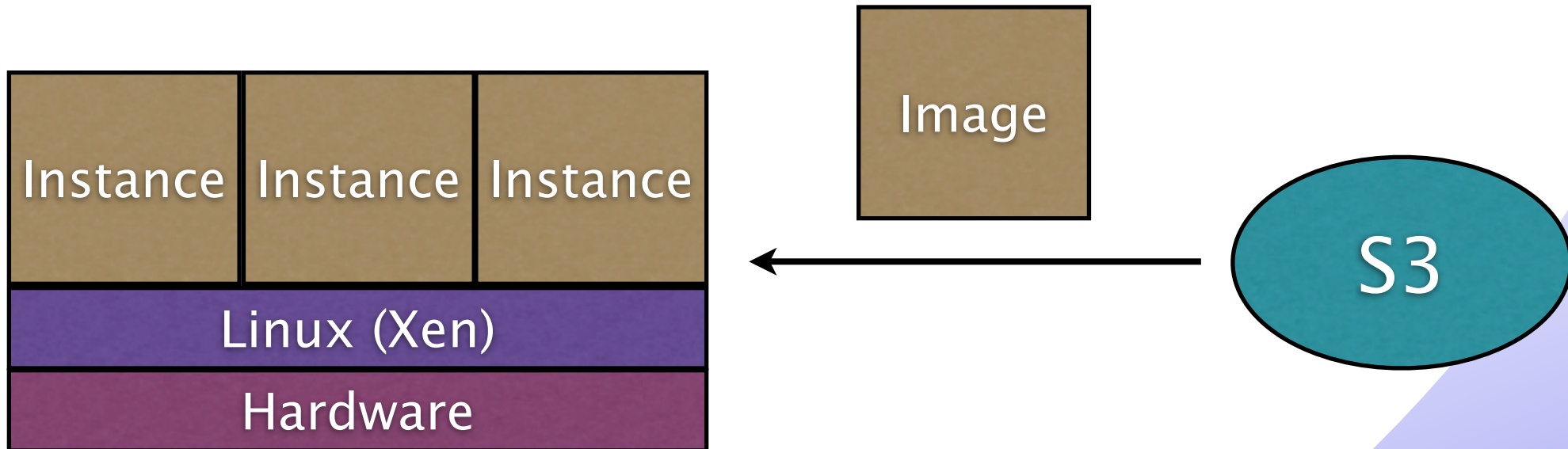


# Dawn Of Utility Computing

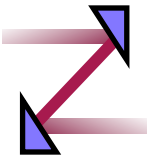
- EC2 and SunGrid
- Bare bones offerings
- Developers unaware
- Software needs to catch up:
  - Architecture
  - Tools etc



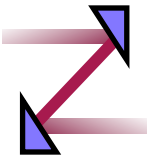
# EC2 Overview



System image (OS + libs etc) transferred from S3, once executing under Xen (VM) becomes an “instance”

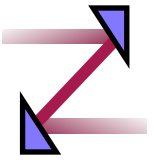


# Challenges



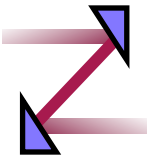
# Where's The Code?

- Where is our software running?
- How does one software component locate another?
- How do we deploy and monitor our software components?



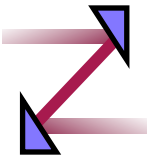
# Storage

- Not what we're used to
- File or Resource oriented
- Centralized databases difficult
- Persistent storage is often a remote service (and slow)
- Local instance storage is often transient (and fast)



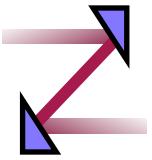
# Naming Service

- DNS is largely useless because name's/ip's change
- Broadcast and Multicast don't work

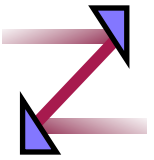


# Configuration

- Must be dynamic
  - Can't be machine specific
  - Can't be defined entirely prior to runtime

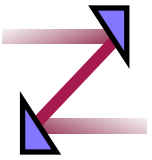


# Baby Steps....



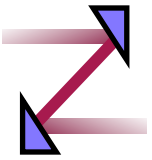
# Naming Service

- Available to all instances
- Can be dynamically discovered and accessed at runtime
- Is a registry of service components



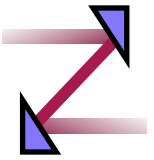
# Deployment Agent

- Allows a single base operating system image with minimal footprint
- One per machine
- Found via Name Service
- Package deploy/undeploy
  - Contain mixture of services and resources



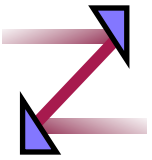
# Service

- Has a lifecycle
- Exposes operational aspects for monitoring
- Can be auto-redeployed on failure
  - Fully de-centralized solution
- Machine-level service to expose operational aspects of machines for monitoring



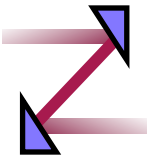
# Exploits Java

- Java is a Platform not a Language
- Downloadable code
- Sandbox environment
- Security



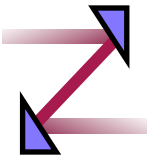
# Next Steps

- Storage
  - Bridge gap between fast and slow
  - Allow it to migrate in face of failure
  - Need to be able to “freeze” it to offline storage at shutdown
  - Some querying concepts
    - MapReduce/Sawzall
  - Hadoop



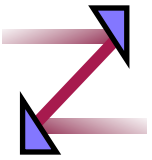
# Next Steps

- Parallel compute problem
  - Graphics work at Brunel
- Made for EC2
- High concurrency potential
  - Antidote to variations in inter-node network latency
  - Got a deadline to meet? Add more nodes



# Next Steps

- Enterprise problem
  - TBD
  - Not as natural a fit as parallel compute problems but possible
  - Dynamic assembly of apps and systems will cause developers some pain



# Utility is Coming

- Cheaper – enterprises starting to pay attention
- Simpler – Dynamic infrastructure with less configuration
- Assumptions – must be few but it's liberating (can't obsess over details)!