Cloud Resilient Architecture (CRA) -Design and Analysis

Glynis Dsouza
Hamid Alipour
Salim Hariri
Youssif-Al-Nashif

Mohamed Eltoweissy

NSF Center for Autonomic Computing
University of Arizona

Pacific National NorthWest Laboratory

Presentation Outline

- Cybersecurity and Cloud Security Challenges
- Experimental Environment and Setup
- Cloud Resilient Architecture (CRA)
  - Moving Target Defense Approach
- CRA Design and Evaluation
- Conclusions

Cybersecurity Challenges

- **Current cybersecurity technologies failed to secure and protect our cyberspace resources and services**
  - **They are mainly signature base, manual intensive and ad-hoc;**
- **According to Phishing Activity Trends Report, data-stealing increased from 36% in 2010 to more than 45% in April 2011.**
- **Cyber attacks can get costly if not resolved quickly.** The average time to resolve a cyber attack is 18 days, with an average cost to participating organizations of $415,748. Results show that malicious insider attacks can take more than 45 days on average to contain.
- **Information theft continues to represent the highest external cost, followed by the costs associated with business disruption**
- **Cyber crimes are intrusive and common occurrences.**
  - **Caused by malicious code, denial of service, stolen or hijacked devices and malicious insiders**

Cloud Security Challenges

- Challenging research problem due to many interdependent tasks

- Reasons for challenge
  - Software Monoculture
  - Dynamic Environment
  - Social Networking

Software Monoculture

- Easy for attacker to study behavior of system and generate attacks
- Vulnerabilities in one software can propagate to a great extent

Cloud Resilient Architecture (CRA)

Software Behavior Encryption

- Spatio Temporal Behavior Encryption
- Moving Target Defense

Self Management

- Automatic Detection and Recovery Capabilities

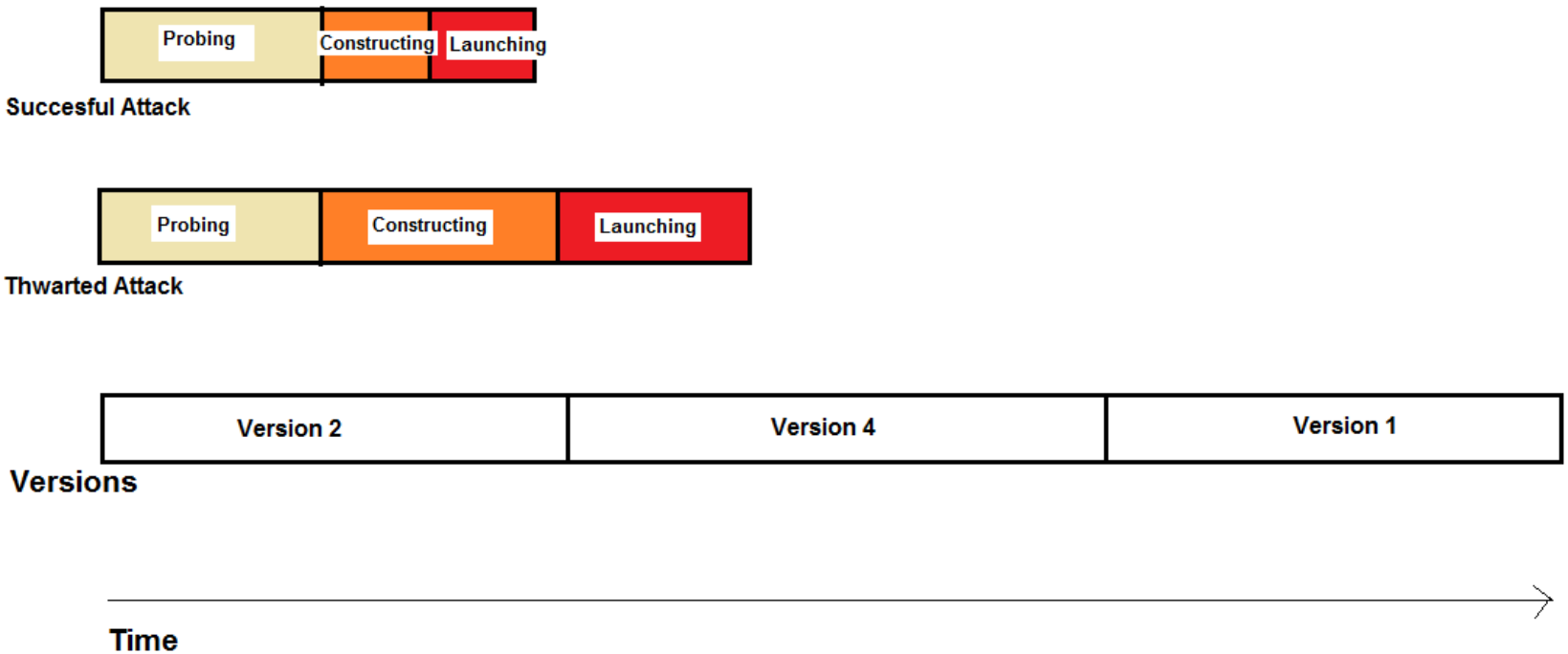
Moving Target Defense

- Vision
  - Create, evaluate and deploy mechanisms and strategies that are diverse, continually shift, and change over time to increase complexity and costs for attackers, limit the exposure of vulnerabilities and opportunities for attack, and increase system resiliency (Source:"CyberSecurity Game-Change Research and Development Recommendations")

Moving Target Defense

- Three Common Techniques of MTD
  - Address Space Randomization
  - Instruction Set Randomization
  - Data Randomization

MTD Attack Lifetime

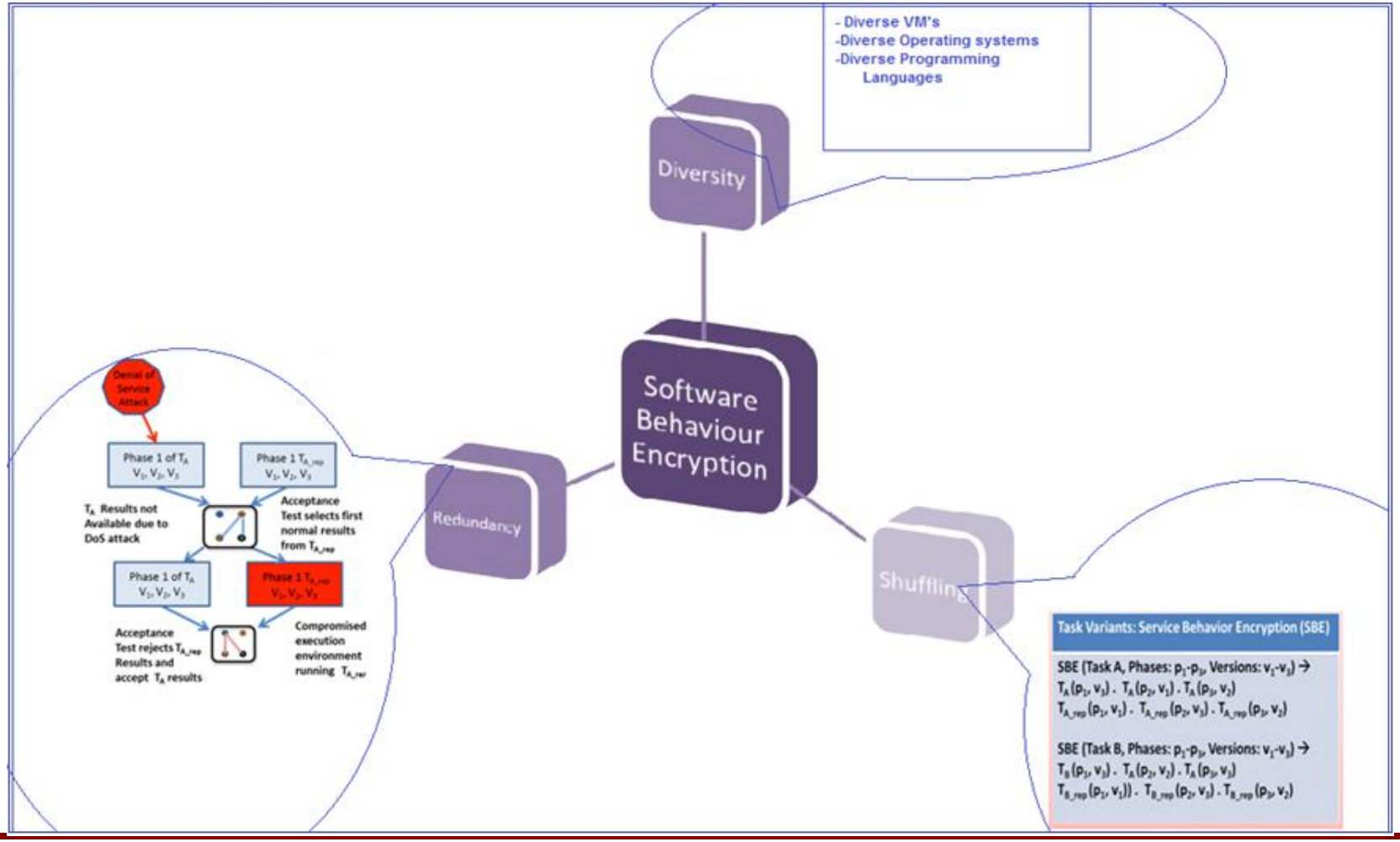


Small time window for attacker

Software Behavior Encryption

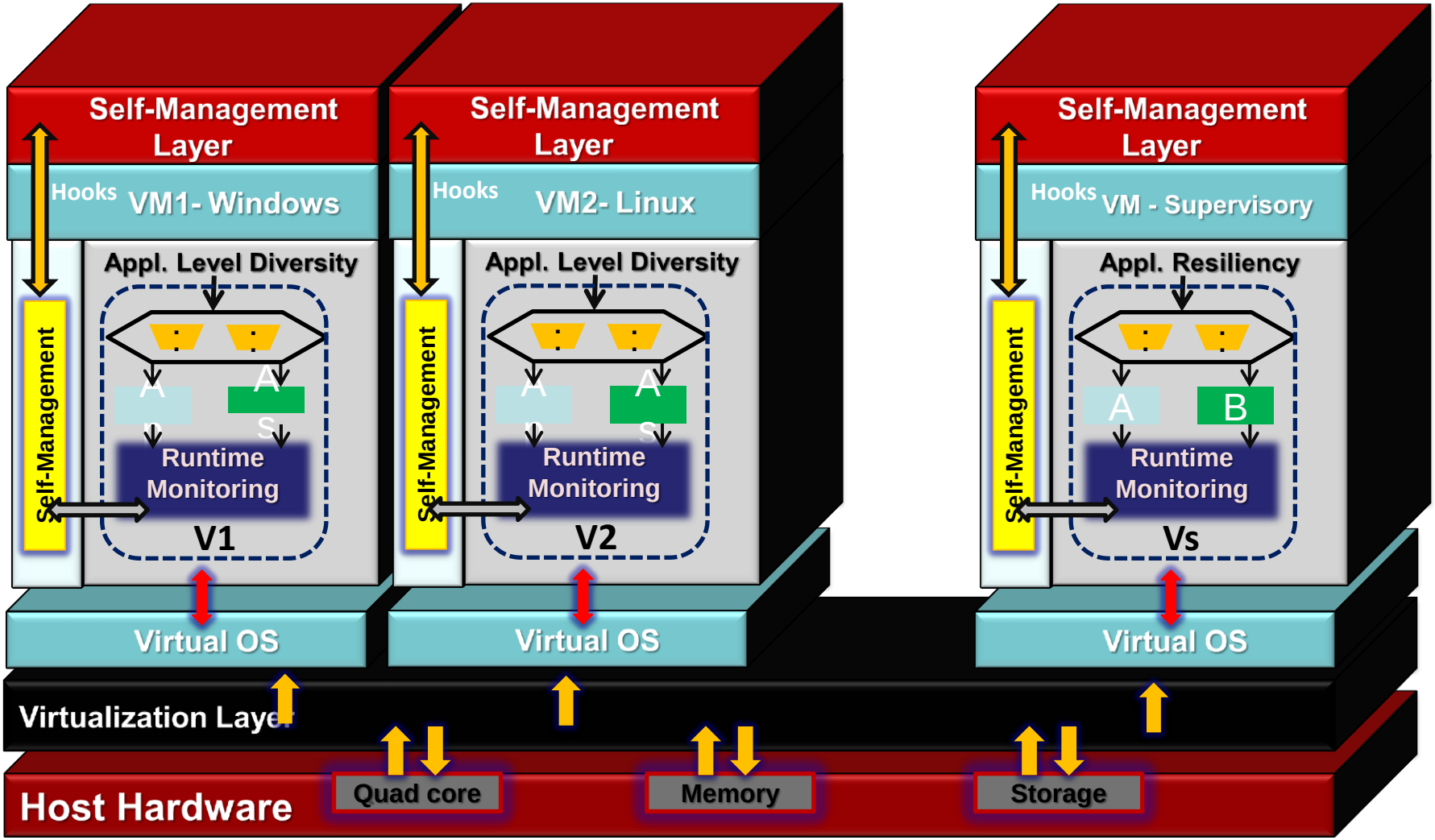
- Diversity
  - Hot Shuffling software variants at runtime
  - Variants are functionally equivalent, behaviorally different
- Redundancy
  - Multiple replicas on different physical hardware
- Shuffling

Software Behavior Encryption



Self Management

```
while true do
    if (any change in management policies) then
        read Controller configuration file and update Controller.policies
    end if
    Currentstate ← Observer.Sensors.getCurrentState()
    Events← Observer.Sensors.getRecievedEvents()
    AnalysisDecision← Observer.analyzeKnowledge(Currentstate,Events)
    if (AnalysisDecision is abnormal) then
        Controller.actionsList=Controller.Policies.check(Events,AnalysisDecision)
        Controller.executeActions()
        LocalEvents=Controller.actionsList.getEvents()
    end if
    if (LocalEvents ≠ null) then
        Controller.reportEventsToPredefinedObservers()
    end if
end while
```


Self-Management Layer
Hooks
VM1- Windows
Appl. Level Diversity
Self-Management
A
A
Runtime Monitoring
V1
Virtual OS
Self-Management Layer
Hooks
VM2- Linux
Appl. Level Diversity
Self-Management
A
A
Runtime Monitoring
V2
Virtual OS
Self-Management Layer
Hooks
VM - Supervisory
Appl. Resiliency
Self-Management
A
B
Runtime Monitoring
Vs
Virtual OS
Virtualization Layer
Host Hardware
Quad core
Memory
Storage

Experimental Environment

- IBM BladeCenter HS22 based Private Cloud
- University of Arizona's Autonomic Computing Lab
- Evaluated on a three node cluster
- Each node has four versions
- Version consists of combination of:
  - Operating System
  - Programming Language
  - E.g. <Linux, C++>, <Windows, Java>

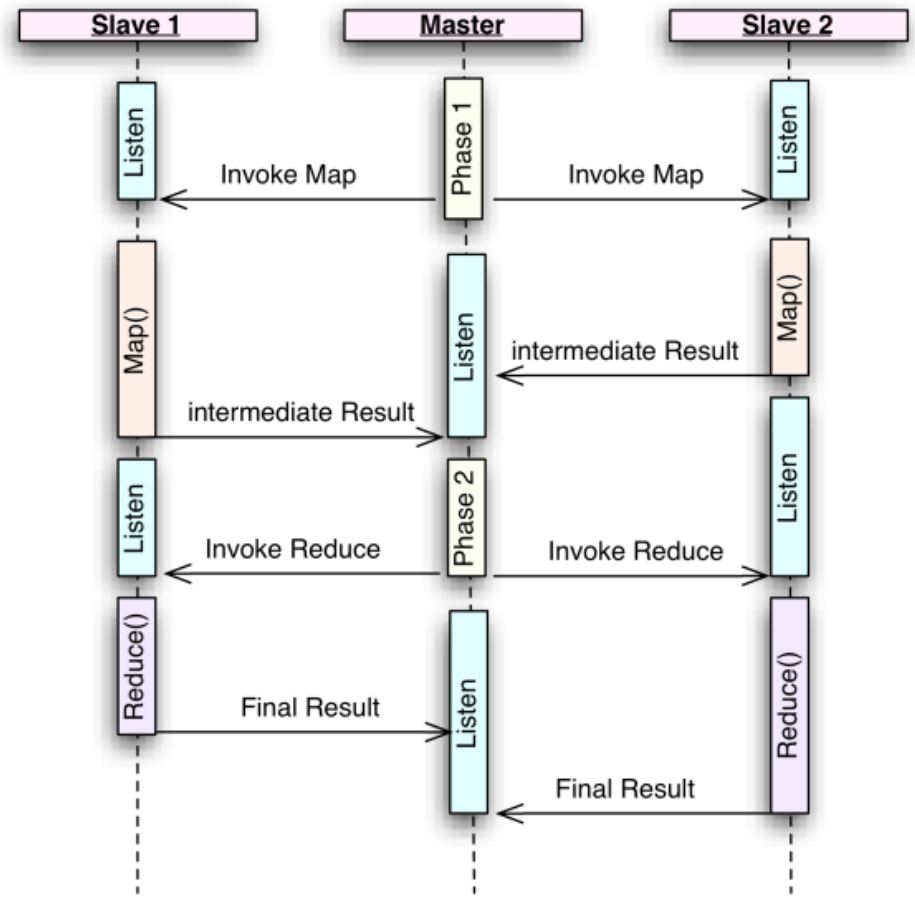
Illustrative Example

- MapReduce Job is divided into three phases
  - First Map on Input 1
  - Second Map on Input 2
  - Final Map/Reduce on above two outputs
- All three nodes run the three phases independently
- Slaves are in Hadoop MapReduce 'Single-Node Cluster' setup
- At the beginning of each phase, a version is selected randomly by each of the three masters.

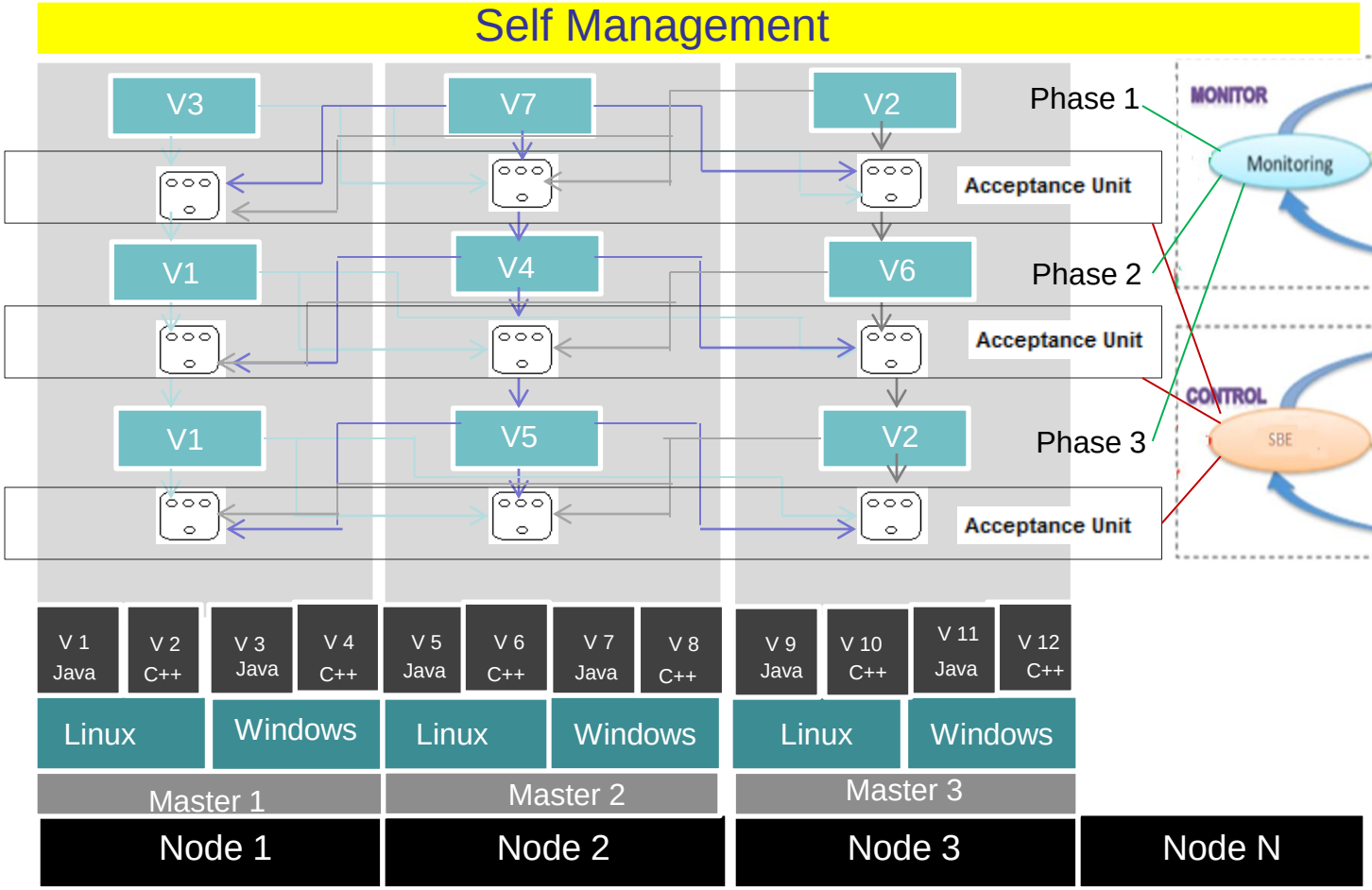
Application used-MapReduce

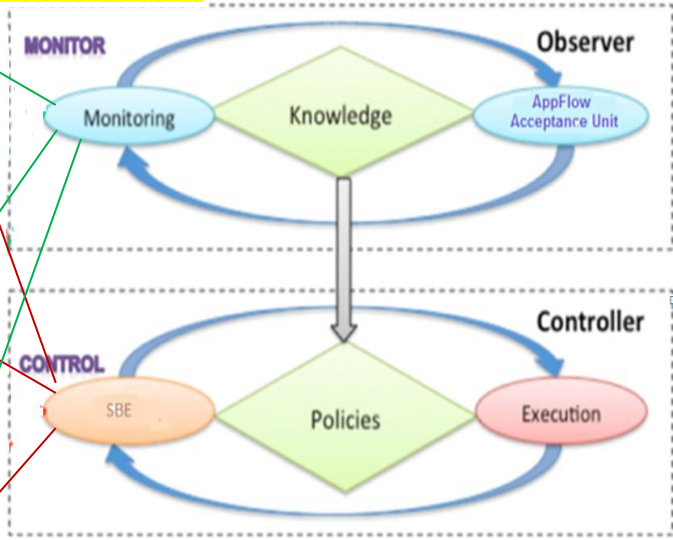
- Large-Scale Data Processing
- MapReduce provides
  - Automatic parallelization & distribution
- MapReduce Wordcount program

MapReduce sequence diagram

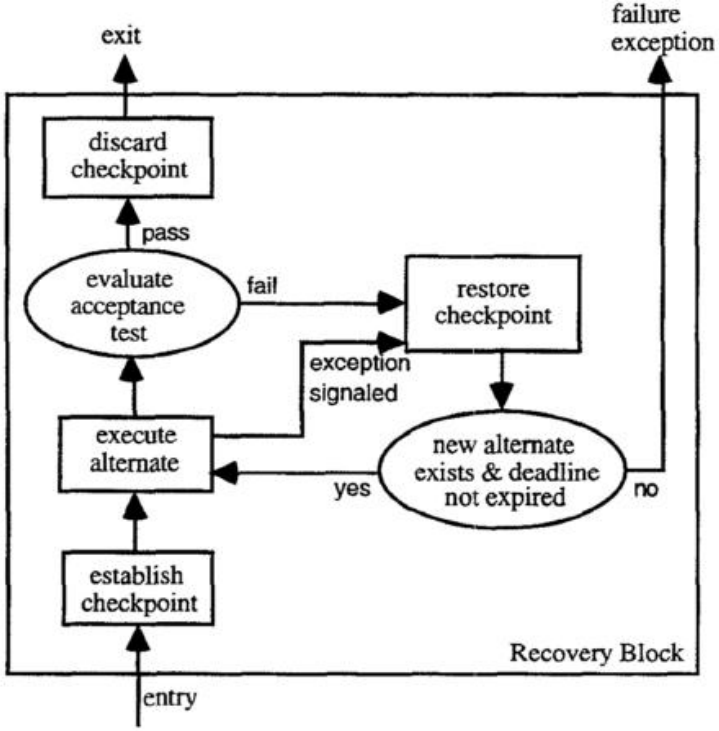


Architecture





Recovery Block - fault tolerance




```
ensure <acceptance test>
by
        (primary block)
else by
        (alternative block 1)
else by
        (alternative block 2)
..............
else by
        (alternative block n)
else    error
```

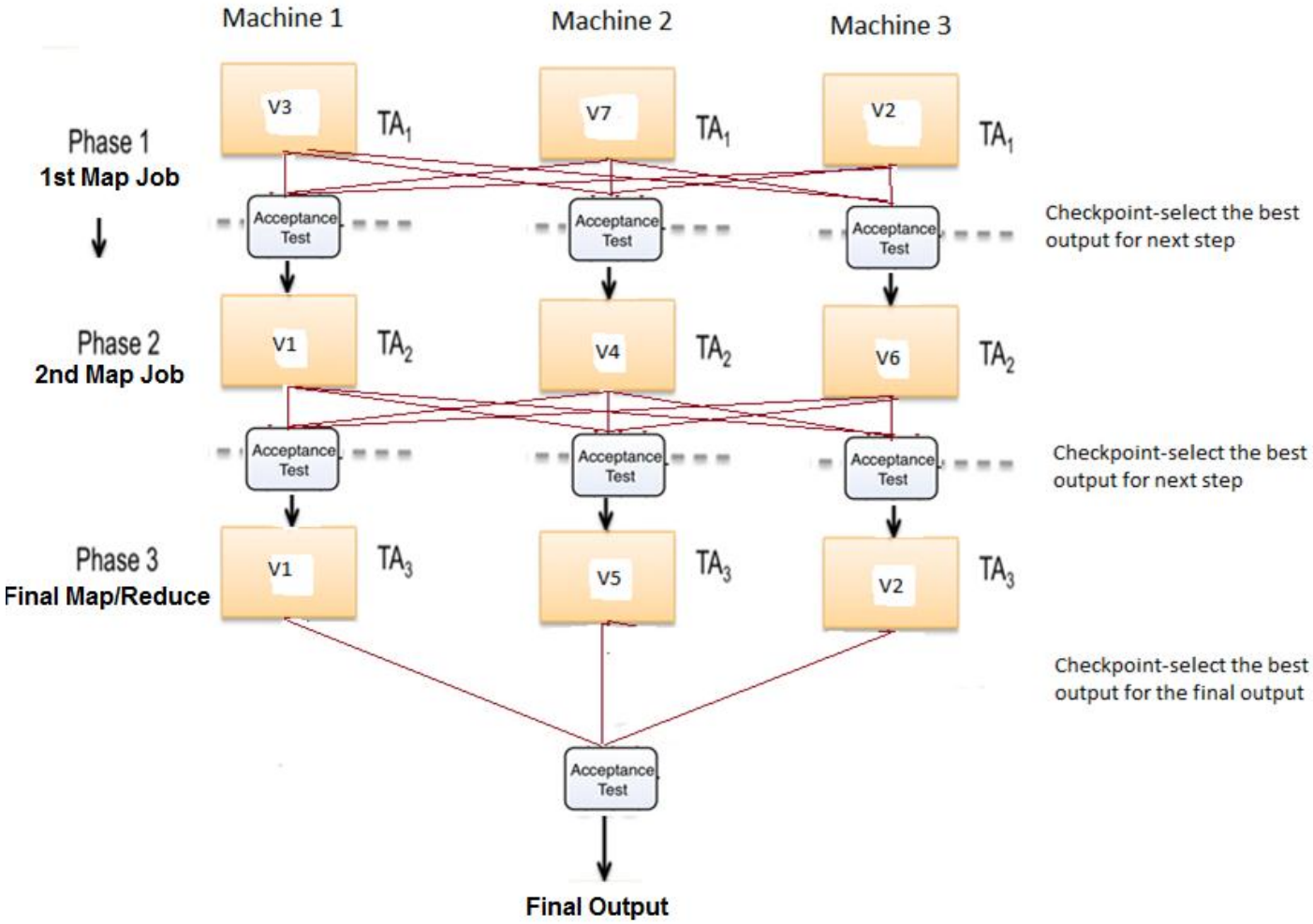
Source: Tyrrell A.;"Recovery Blocks and Algorithm Based Fault tolerance ";
1996. Conference Proceedings of the 22nd EUROMICRO Conference

- Checkpoint based recovery technique

Acceptance test

- At the end of each phase, results are given by the slave to the Master
- Master runs local acceptance test
  - Output of each Map phase is in the form of <keyword 1>
  - Output of the final Map/ Reduce phase is in the form of <keyword count>
  - Any other outcome is considered as failure
- If the acceptance test fails, output is requested from the other masters

Acceptance Test



Task Management Algorithm

```
Initialize SBE
Controller.generatePolicies(SBE.getTasksAndPhasesVersionList())
Observer.setTasksPhaseSensors()
Observer.startTasksPhaseSensors()
for (phase=1 to SBE.numberOfPhases) do
  exitFlag←0
  for (task=1 to SBE.numberOfReplicas) do
    Controller.launchTask(task,phase);
  end for
  while (exitFlag=0) do
    while (true) do
      state←Observer.getAllTasksCheckState(phase)
      if (state='A task finished this phase successfully') then
        Controller.killThisPhaseForAllTasks(state,phase)
        Controller.updateAllTaskReplicasWithTheSuccessfulOutput()
        exitFlag←1
        breakLoop()
      else if (state='A task is running abnormaly') then
        taskWithAbnormality←state.getAbnormalTask()
        Controller.killAbnormalTask(taskWithAbnormality,phase)
        SBE.reassignPhasesVersionsForAbnormalTask(taskWithAbnormality,phase)
        Controller.relaunchTask(taskWithAbnormality,phase)
      end if
    end while
  end while
end for
```

Case 1: Resilience against Dos Attacks

Denial of Service attack on Windows VM-6

	Response Time (in seconds)	
	Without DoS attack	**With DoS attack**
Without MTD	95	615
With MTD	105	105

Case 2:Resilience against Insider Attacks

Compromise attack on Linux VM-1

	Response Time (in seconds)	
	Without Insider attack	**With Insider attack**
Without MTD	95	No response
With MTD	105	105
% increase in response time with MTD		11%

Conclusions

- We cannot build perfect cloud security systems,
- CRA based on MTD provides a promising approach to build intrusion tolerance Cloud Services
- Anomaly behavior analysis will help driving when to change the environment and respond optimally to attacks or malicious events.
- Autonomic computing (Self Management) can be the underlying paradigm to implement Cloud applications and services

Questions??

Thank You