Information Operations & Security
(Dr. R. L. Herklotz, 5 August 2013)
Assured Cloud Computing

Roy Campbell,
Indy Gupta, José Meseguer, Rakesh Bobba, David Nicol, Ravi Iyer,
Zbigniew Kalbarczyk, Bill Sanders, Gul Agha

United States of America
Department of the Air Force
http://www.airforce.com

University of Illinois at Urbana-Champaign
http://www.illinois.edu

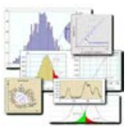
Information Trust Institute
http://www.iti.illinois.edu

Innovative Use of Advances in Computing

Cloud computing infrastructure

Robust computing at low-cost , “pay-as-you-go”

- Analysis



- Integration

- HMI

Assuring security and safety of the nation

United States Air Force
global vigilance, reach and power

Smarter Planet Ecosystem Solutions which are:

Cost effective
Environment friendly
Trustworthy

Large volume of data

Phones, Sensors
Smart cars

Individuals & enterprises

Human expertise
Innovations
Education
Research

Benefits to individuals & society

Modern health care
Adaptive Power Grid
Efficient transportation (air, ground, sea)
Preservation of water
New age agriculture

Emerging Concern: Big Data a major bottleneck

- **Big Data Problem**
  - Scientific invention
  - Engineering
  - Sensing
  - Computational Genomics
  - Manufacturing
  - Agriculture
  - Politics and strife
- **Needed: Engineering Discipline to process Big Data**
  - Application data intensive algorithms and data structures
  - Middleware (Data storage – beyond Hive, Giraffe, MapReduce, Storm…)
  - Data intensive architectures

Grainger Big Data Initiative

- $35 million for initative
- 13-15 New senior faculty chairs and professorships
- Backing of University of Illinois
- Goal – to create effective discipline of engineering big data systems
  - What are the disciplines topics?
  - What are the measures of success?
  - Who are the partners?

Research Results at Assured Cloud Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Research Results at Center

- **Indy Gupta - Real Time Challenges in Clouds**
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

The Challenges

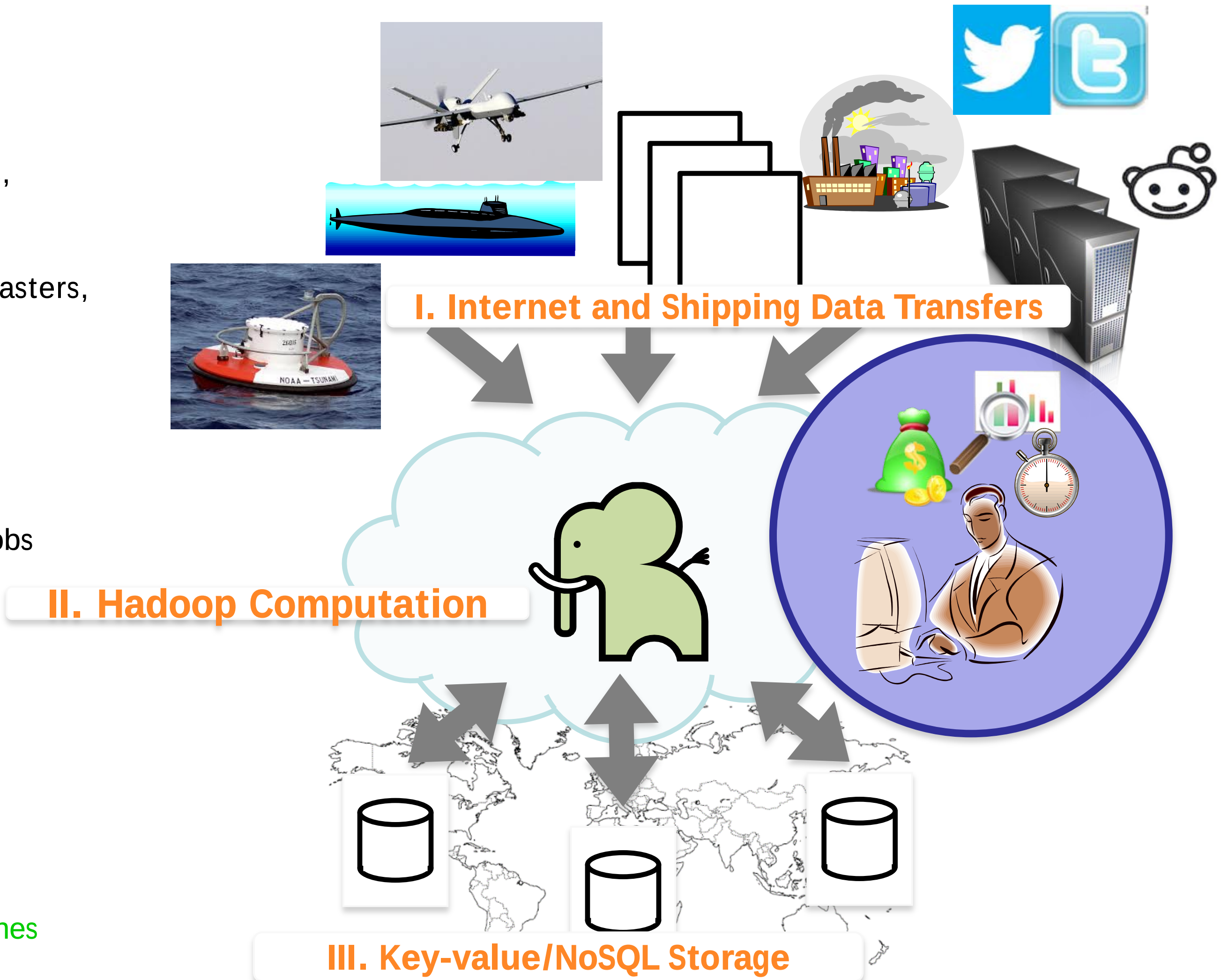
- **Preparedness, Rescue and Recovery** requires:

I. Data Transfer to cloud
- From ocean sensors, UAVs, vehicles, humans, social networks, etc.
- → Internet and, under disasters, shipping physical disks

II. Computation in cloud
- → Hadoop ($14B by 2017)
- For command and control visualization
- High priority jobs (e.g., jobs dealing with **rescue**)
- Lower priority jobs (e.g., **recovery**, other jobs)

III. Data Storage in cloud
- Fast ops → NoSQL storage systems ($3.4B by 2018)

- **Constraints**
  - Hard real-time job deadlines
  - $ (Budget) on transfers
  - Overloaded Clusters and Network



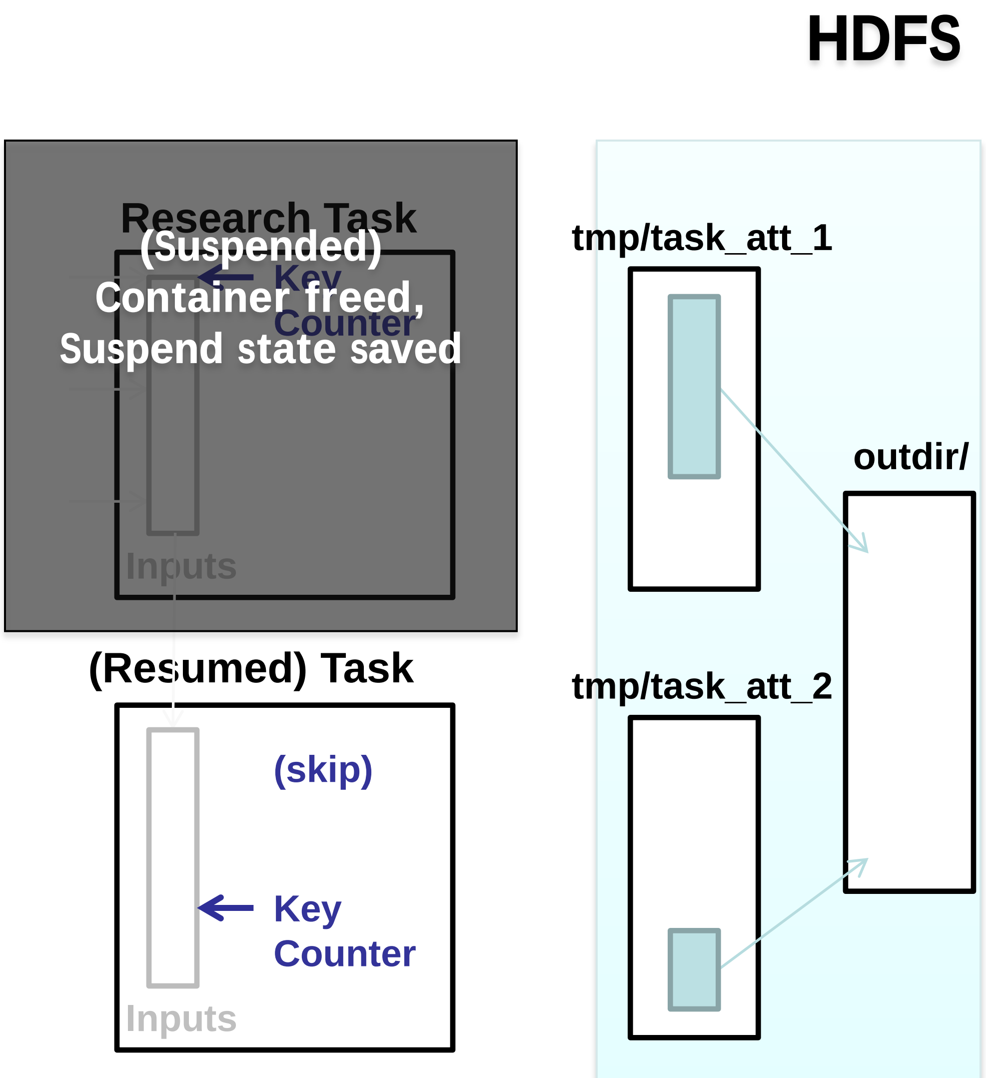
Contributions

Our System	Application-specified Constraint	Also Optimizes	Target Scenario
I. Internet and Shipping Data Transfers			
Pandora-A	Deadline	Low $ cost	**Rescue, Recovery**
Pandora-B	$ Budget	Short transfer time	**Preparedness**
II. Hadoop Computation			
Natjam	Support job priorities	Job completion time	**Rescue, Recovery**
Natjam-R	Support real-time deadlines	Job completion time	**Rescue, Recovery**
III. Key-value/NoSQL storage			
Model-checking of Cassandra NoSQL system	Data Consistency	Fast operations (Availability)	**Preparedness**

More info? Visit http://dprg.cs.uiuc.edu

The Natjam System

- High priority jobs = Production jobs (rescue)
- Low Priority jobs = Research jobs (recovery)
- All batch jobs only
- Jobs consist of parallel tasks (e.g., Map tasks, Reduce tasks)
- Today's Hadoop clusters support job priorities by either
  - Having production jobs wait for scheduled research tasks to finish → Prolongs production jobs ☹
  - Killing tasks of research jobs → Prolongs research jobs ☹
- Our system – Natjam
  - Built directly into Hadoop YARN (0.23)
  - Research tasks can be evicted immediately, but they save a fast on-demand checkpoint
    - Checkpoint contains only Key counter, and no other state
  - When restarted, research task can resume from checkpoint



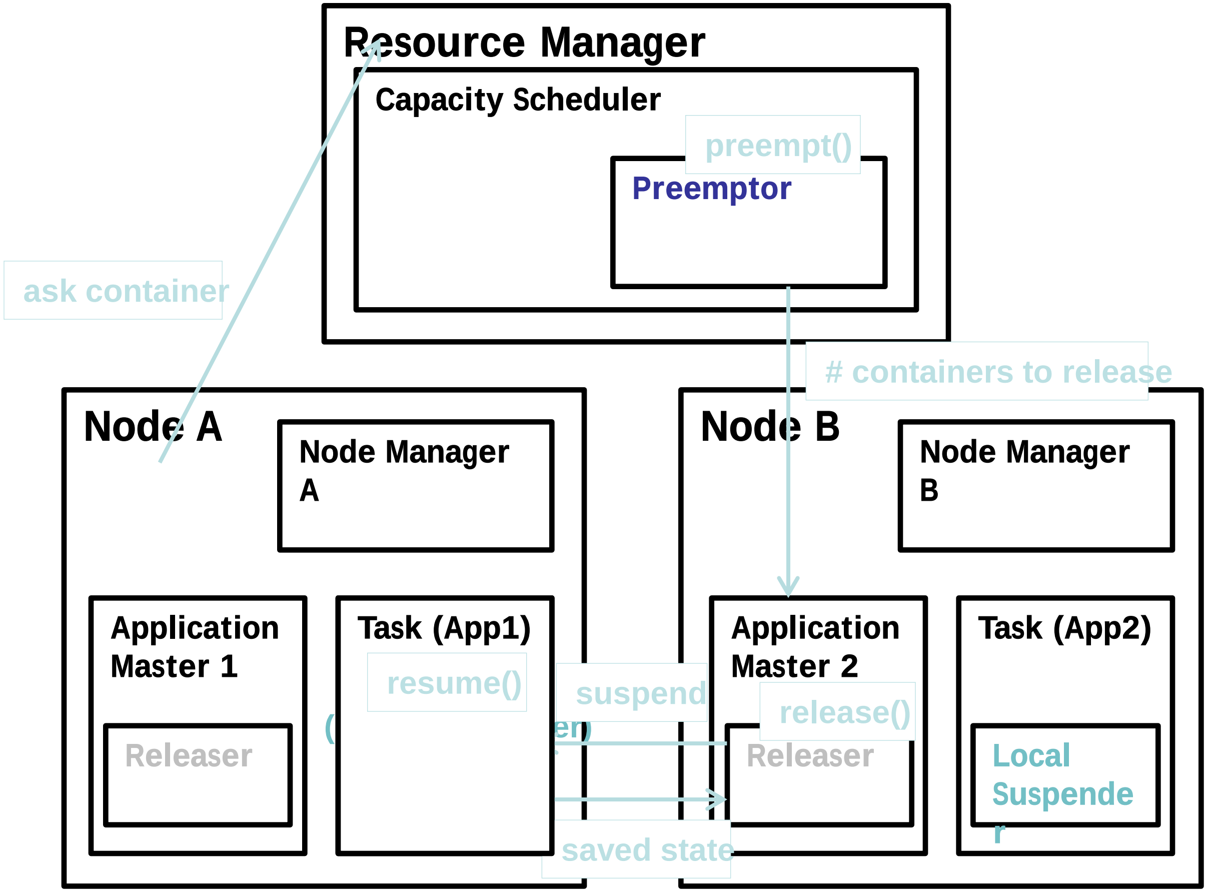
Evictions in Natjam

Natjam uses Smart eviction policies

- **Job Eviction Policies**: Resource-based (e.g., Most allocated resources = MR).
- **Task Eviction Policies**: For tasks within victim job. Time-based (e.g., task with shortest time remaining = SRT).
- Evict only reduces, since reduces are longer than maps (231 s vs 19 s [Facebook])

Natjam-R Extension

- For jobs with hard real-time deadlines
- Maintain one queue in Hadoop
- Job Eviction Policies: Evict jobs with larger deadline



More info? Visit http://dprg.cs.uiuc.edu

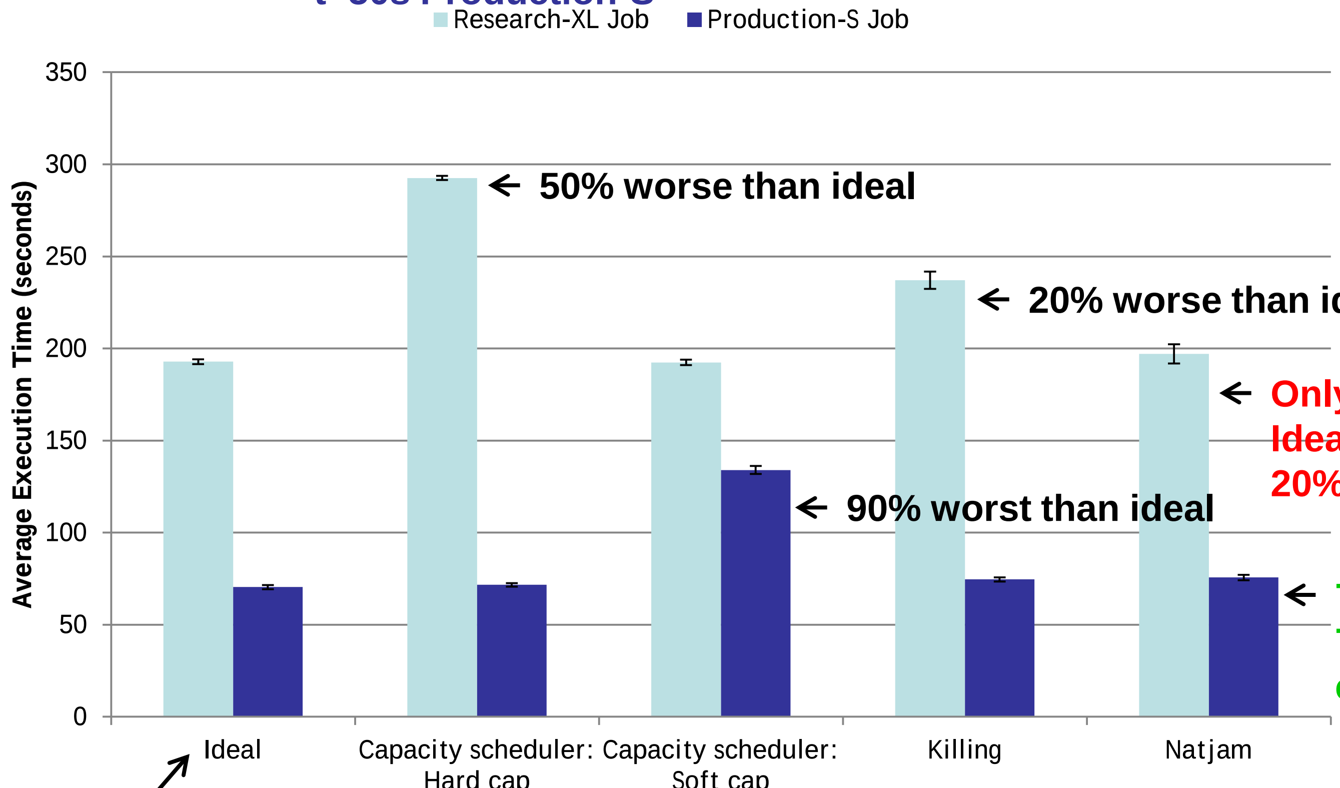
Natjam: Production and Research Jobs

time (seconds)

t=0s Research-XL

t=50s Production-S

(7 server cluster from CCT)



Only 2% worse than Ideal;
20% better than Killing

7% worse than ideal
77% better than Soft cap

Empty Cluster

More info? Visit http://dprg.cs.uiuc.edu

Interesting Results, Wrap Up

- Pandora Planner available live at: http://hillary.cs.uiuc.edu/
- Natjam proven to work via extensive experiments on 250-server Yahoo! cluster and using Hadoop traces from Yahoo's commercial clusters
- **Natjam**
  - Shortest Remaining Time eviction policy is optimal
    - Counter-intuitive since it is longest task first scheduling policy (shortest task first scheduling is optimal in multiprocessor systems)
- **Natjam-R**
  - Deadline-based eviction policies better than resource-based policies
    - Latter are more "fair" to jobs, and miss many deadlines; former at least meets some deadlines
- **Next Steps**
  - Support chains and DAGs of Hadoop jobs with deadlines
  - Integrate Pandora transfer system and Natjam computation systems

More info? Visit http://dprg.cs.uiuc.edu

Research Results at Center

- **Indy Gupta** - Real Time Challenges in Clouds
- **Jose Meseguer - Model-checking NoSQL Storage Systems**
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Why NoSQL Storage?

- **Availability**: Fast read and write operations
  - Essential **in Rescue and Recovery** scenarios
- NoSQL systems are three orders of magnitude faster than RDBMs like MySQL
- The catch: NoSQL systems offer weaker models of **Data Consistency** than ACID
  - E.g., Eventual Consistency on server side
  - Reads at client *might* return stale data
  - **Staleness affects correctness at clients, esp. in rescue/recovery scenarios**
- Why? CAP Theorem: No storage system can achieve both Strong (ACID) Consistency and Availability
- NoSQL systems (includes key-value stores) are growing quickly and projected to be a **$3.4B industry** by 2018

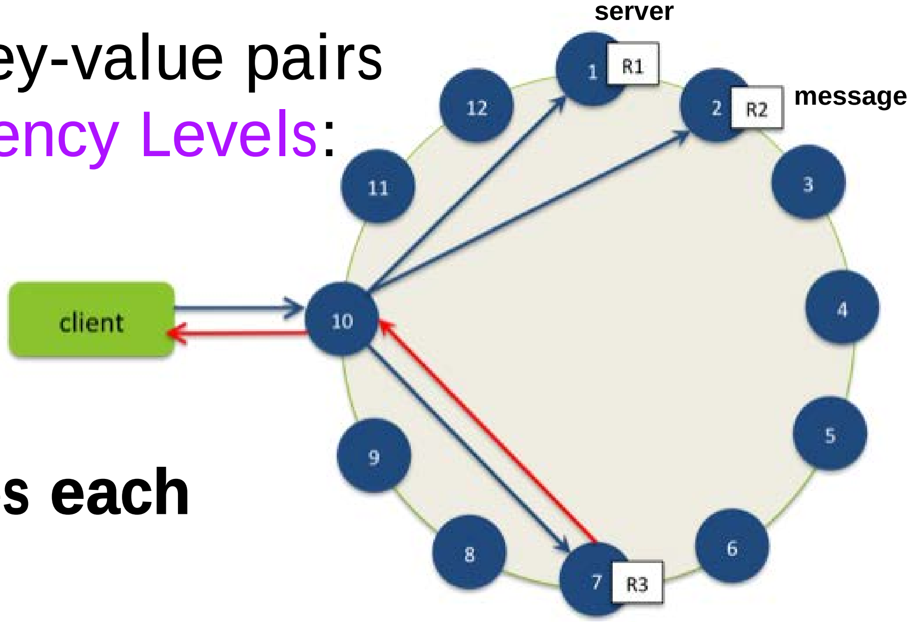
Verifying Client-Side Consistency in NoSQL

- *Does a given NoSQL system really meet certain client-side consistency models?*
  - *Do reads return stale data?*
  - *How stale is this data?*
  - *Under variable message delays and failures?*
- *Important for **Preparedness***

- **Our First Target: Cassandra NoSQL System**
  - **Open-source, developed by Facebook, widely used**
- **The Tool: Maude model-checking system developed by Prof. Meseguer's group**
- Why model checking?
  - Comprehensive coverage of state space
  - Independent of implementation choices (e.g., language) or optimizations
  - Can build towards a science of design

Cassandra From 30K Feet

- Cassandra cluster stores key-value pairs e.g., key=sensor id, value=reading
- Each key-value pair replicated at multiple servers
- Clients can read/write key-value pairs
- Cassandra offers Consistency Levels:
  - Each client individually can specify how many servers need to answer
  - E.g., All, Quorum, One

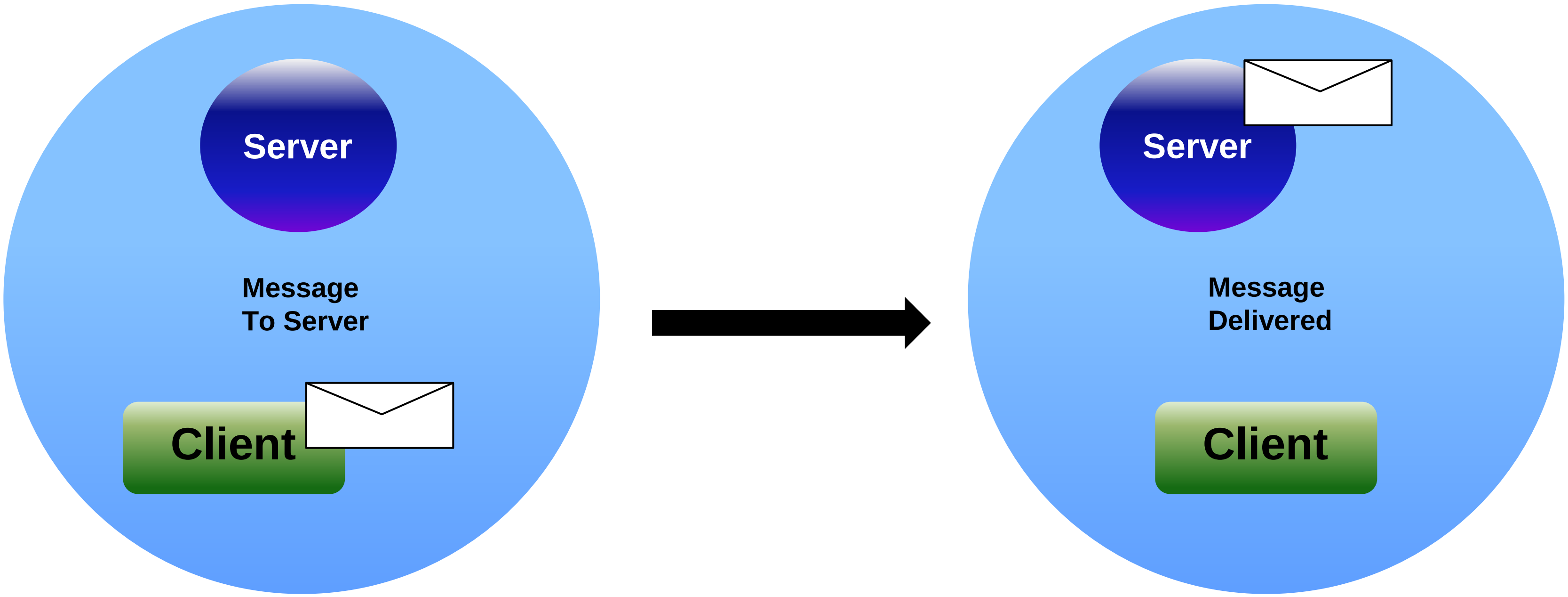


Our Maude model captures each server, client, message

Many details of system behavior that need to be captured

Intuition: Basic Rule in Maude

- Rules describe how messages transform recipients



- Maude explores the state space of all reached global states
- Verifies consistency violations in each state

Results

Read-Your-Own-Writes Consistency Model: Violations

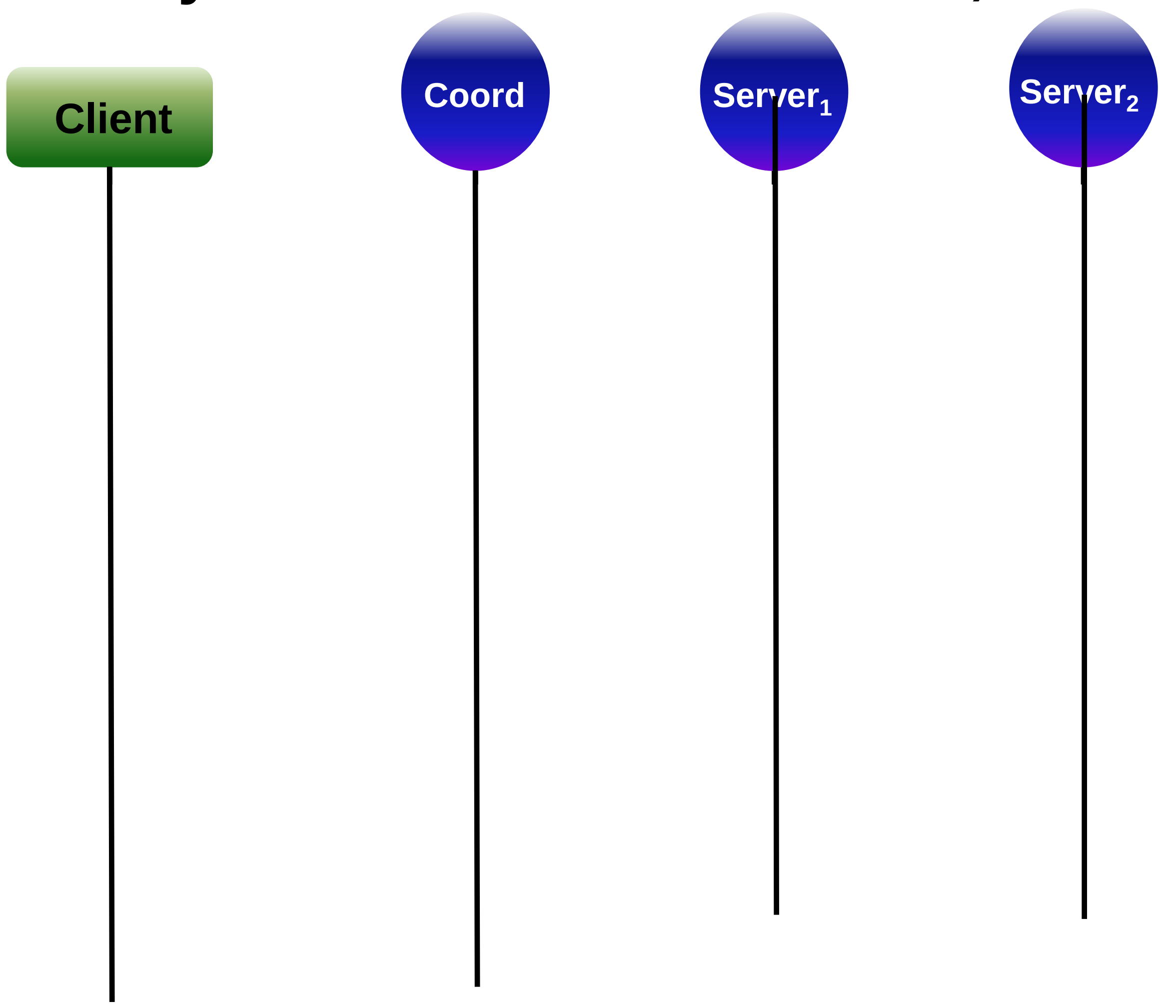
Write_1 \ Read_2	One	Quorum	All
One	✗	✗	✓
Quorum	✗	✓	✓
All	✓	✓	✓

Cassandra's consistency levels

Conclusion: Different clients using different consistency levels can cause global inconsistency

Timeline

Consistency Violation Discovered During Analysis
Consistency Levels used: Read=ONE, Write=ONE



Violation occurs because read reaches server before write

Next Steps

- Modular components for design of NoSQL systems
  - Designers can pick and choose, depending on deployment requirements
- Science of Design for NoSQL systems
  - Independent of implementation choices, e.g., language
- Characterizing the Consistency-Availability Tradeoff Space
  - How close are today's systems to the achievable boundary?
  - Can we get closer?
- Finding Bugs in today's NoSQL systems
- Other target NoSQL systems: RIAK, MongoDB, VoltDB

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- **Rakesh Bobba - Group Key Management as a Service**
- **Jose Meseguer** - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Why Group Key Management as a Service?

- Many cloud applications involve distributed computation with communications among multiple processes
  - joint search and rescue operations may involve collaborative processing of sensor data
- High assurance cloud computing demands that these communications be protected.
- Key management is an important security function underpinning secure communications
  - Group key management supports secure communication among groups (e.g., people, processes)

Problem

- Group key management systems
  - Distributed/Decentralized: too complex with high overheads
    - e.g., FDLKH, TGDH, DLPKH
  - Centralized: simple but with single point of failure
    - e.g., LKH

- Can we design a simple group key management service reliable and scalable enough to meet cloud computing needs?

Approach

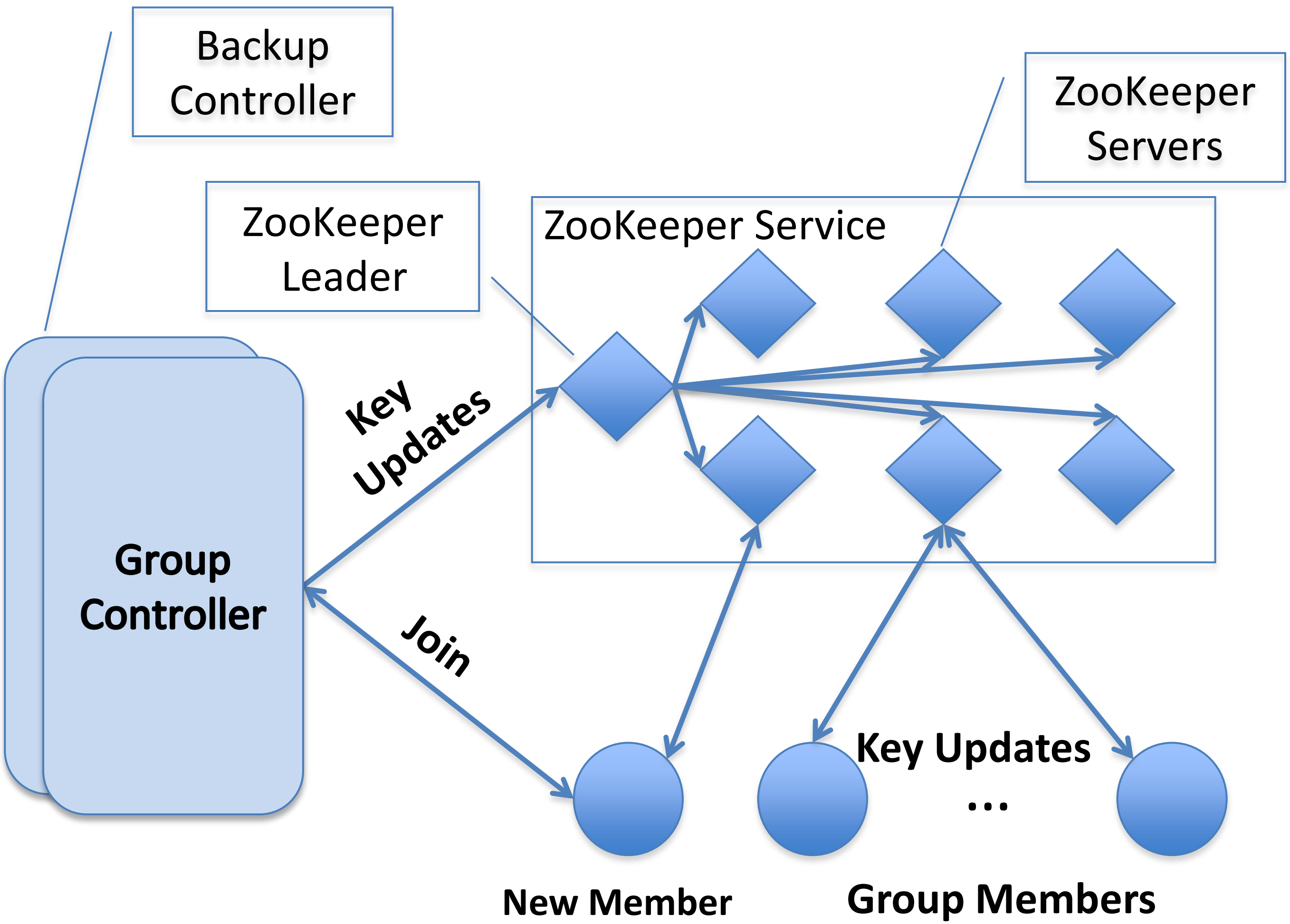
- Designing a fault-tolerant and distributed service can be error-prone
  - Use commercial-off-the-shelf (COTS) or commonly available coordination services as the basis of our distributed framework
- Testing such designs through implementations is expensive
  - Formally model and analyze the design, to quickly test and refine it without have to build it
- Specifically:
  - We use the ZooKeeper distributed coordination framework as the basis for a group key management service design
  - We use Maude and PVeStA – a parallel statistical model checking tool – to model and analyze the reliability and performance of the design

Approach

- Our design consists of:
  - A group controller who authenticates clients and generates group keys
  - A ZooKeeper instance in the cloud which securely stores and broadcasts group keys, and stores the state of the group controller
  - Clients who wish to receive group keys from the key management service

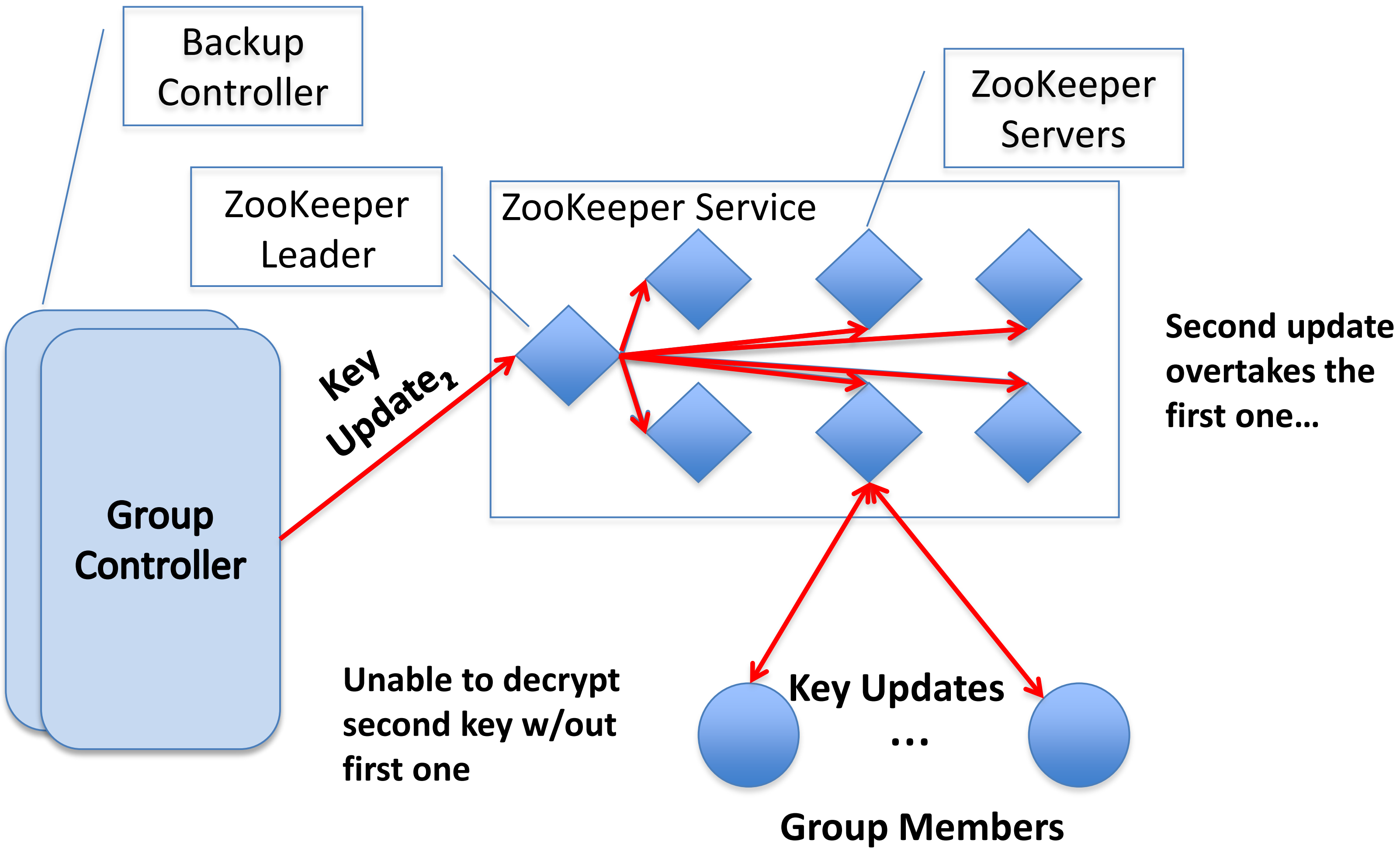
Approach

Design Overview: ZooKeeper-based Group Controller And Joining Client



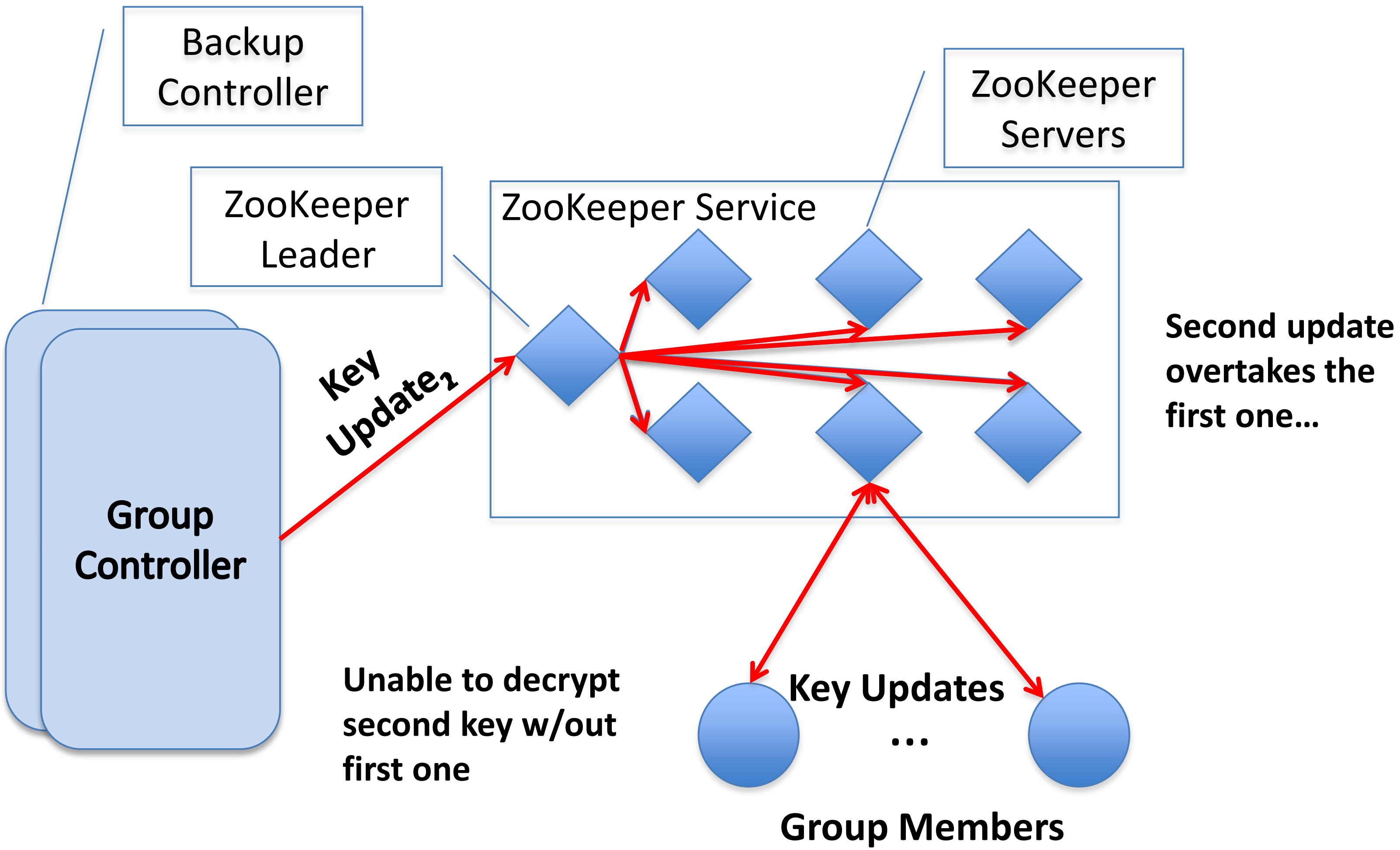
Preliminary Results

Using our Maude and PVeStA based modeling and analysis we uncovered a flaw in our design



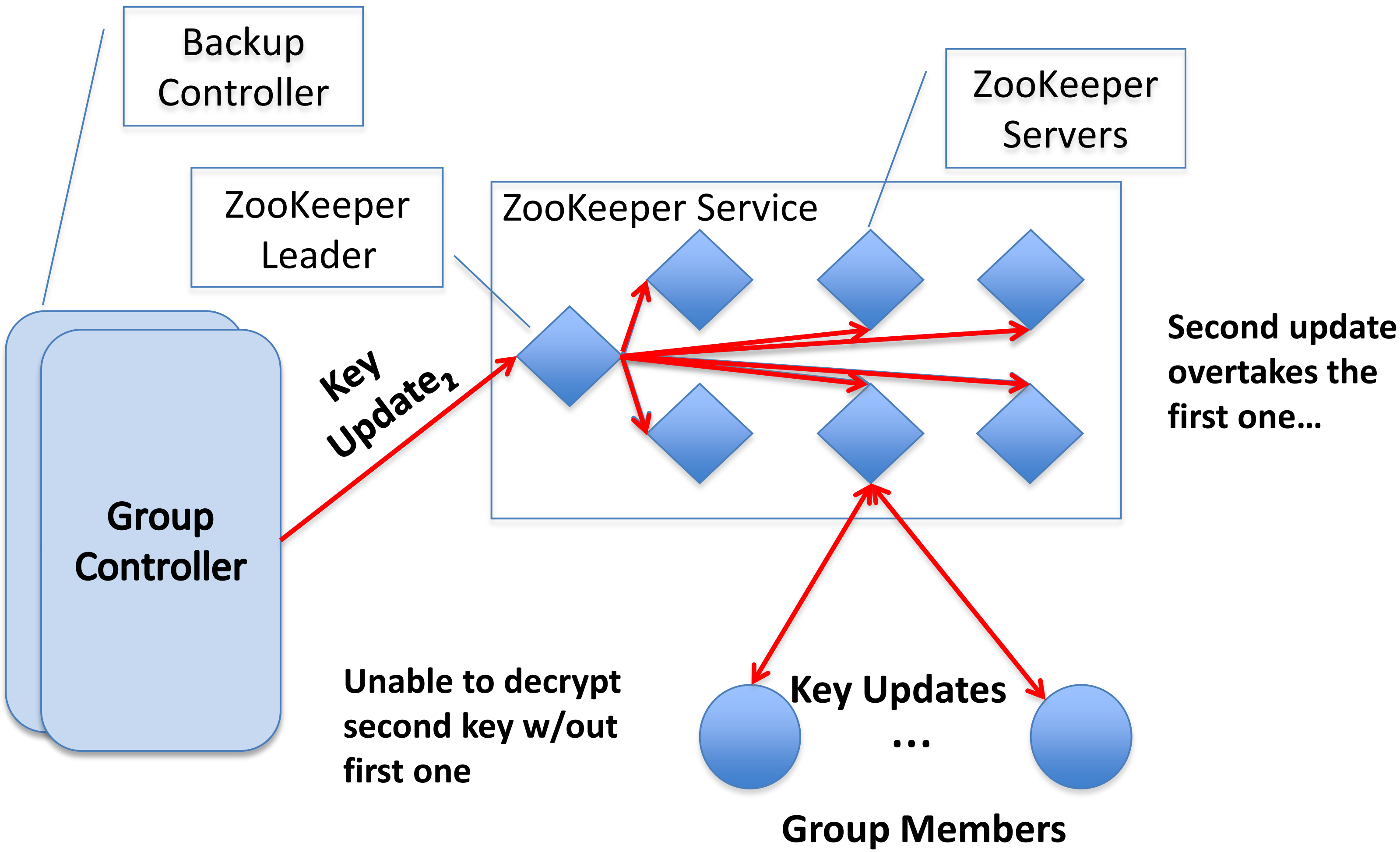
Preliminary Results

With this flaw our experiments showed that only 92% - 96% of key updates were reaching clients even when there are no server failures.



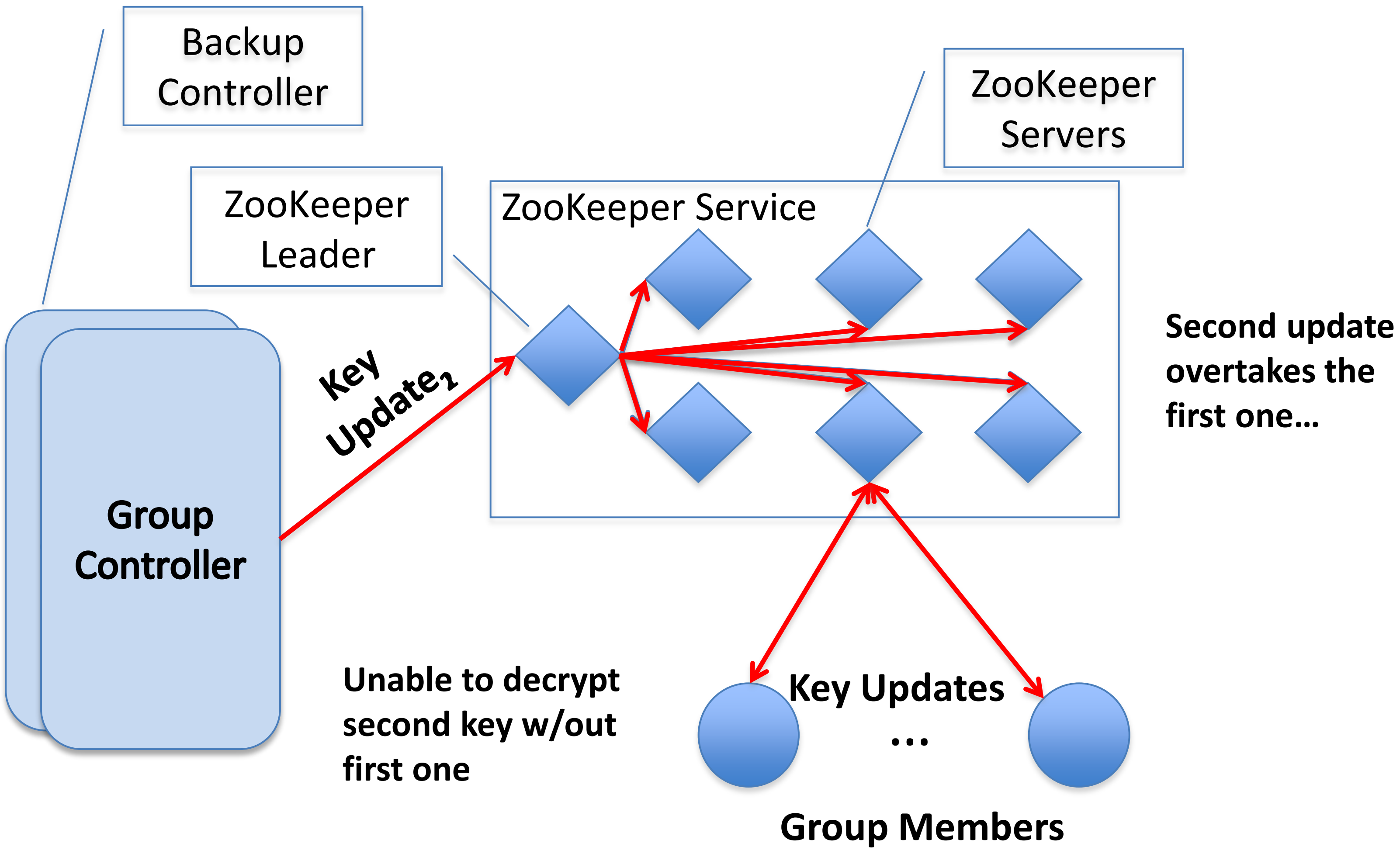
Preliminary Results

Fix: Need to ensure sufficient time between key updates operations (regular, join, leaves) to allow for key propagation – e.g., batching



Preliminary Results

With the fixed design 100% of the key updates were received by clients.



Next Steps

- More extensive experiments to better characterize the performance and reliability of the design
- Extend failure model to include network failures
- Extend to more sophisticated group and other key management schemes

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- **Rakesh Bobba** - Group Key Management as a Service
- **Jose Meseguer - VA Admission Control**
- David Nicol - Quantifying Trust in the Cloud
- **Roy Campbell** - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Motivation

- Cloud Infrastructures
  - Provide flexible and elastic compute power to users
  - Users can instantiate their own application servers with tailored software configurations – Virtual Appliances (VAs)
- For High Assurance Cloud Infrastructures (e.g., Air Force Private Cloud)
  - Need to ensure the software integrity of VAs to reduce the risk to the integrity of the infrastructure
    - In joint search, rescue and recovery operations where VAs may come from partners

Problem

- Ensuring the integrity of VAs - need admission control and monitoring of VAs
- Admission Control (static & offline)
    - Ensure VAs contain what they say they contain (e.g., legitimate software) and not something else (e.g., viruses, malware)
        - compliance with admission policy
    - Malware or virus scanning is necessary but not sufficient
- Monitoring (run-time & online)
    - Detect when a VA launches unauthorized or unexpected software

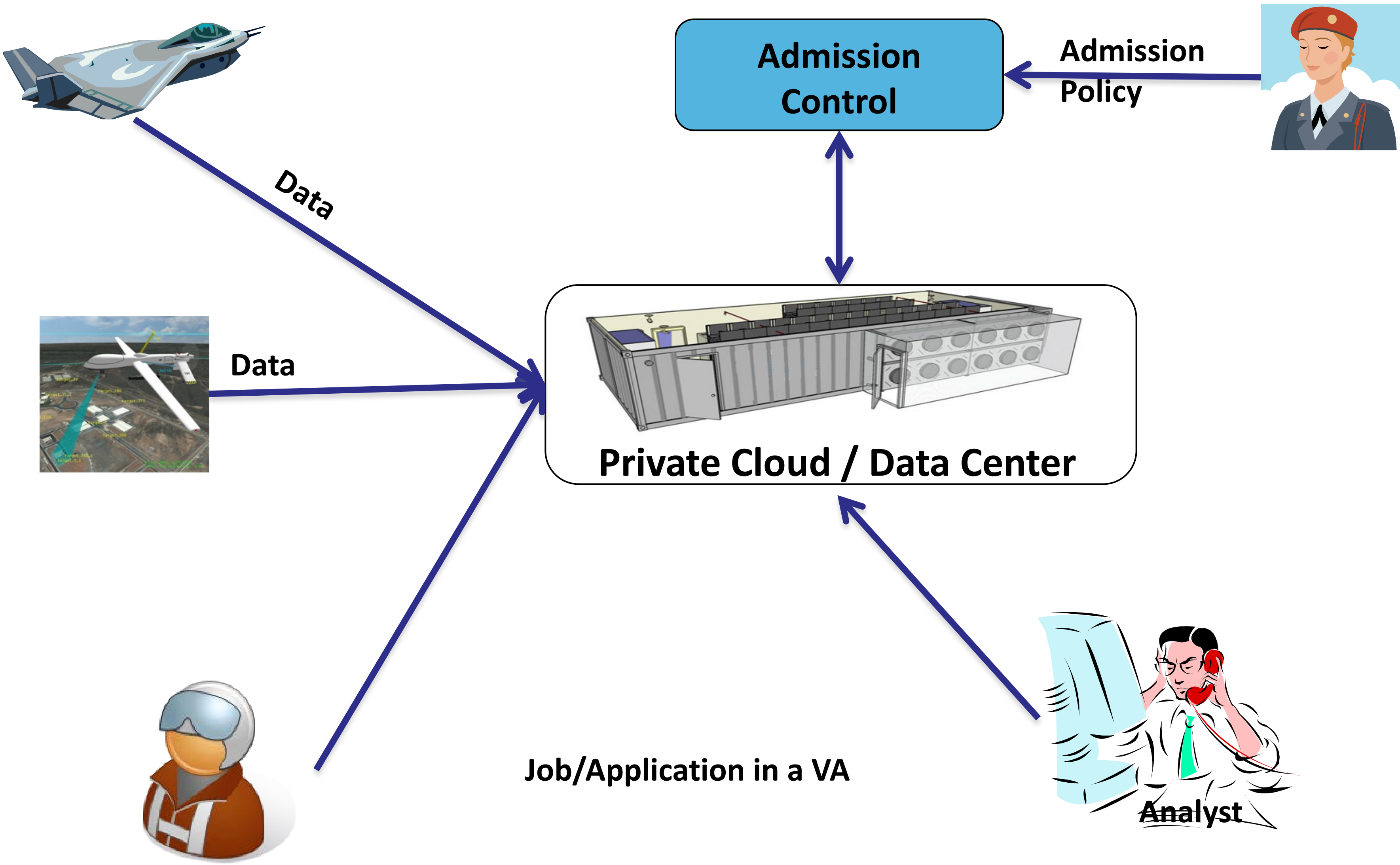
Problem

- Ensuring the integrity of VAs - need admission control and monitoring of VAs
  - Admission Control (static & offline)
    - Ensure VAs contain what they say they contain (e.g., legitimate software) and not something else (e.g., viruses, malware)
      - compliance with admission policy
    - Malware or virus scanning is necessary but not sufficient
  - Monitoring (run-time & online)- Ravi and Zbigniew's group
    - Detect when a VA launches unauthorized or unexpected software

Approach

- We propose a whitelist-based framework to complement blacklisting approach exemplified by virus and malware scanning
- Software in VAs, at the file level, is checked against whitelist of known-good hash values
  - hash values can be obtained from software publishers
- Based on the *presence* of unverified files and missing files, we rate software integrity
  - **3** "fully verified" and integrity protected
  - **2** "partially clean" or medium integrity
  - **1** "modified" or low integrity
- Admission policies can then be based on the rating

Framework Use Case

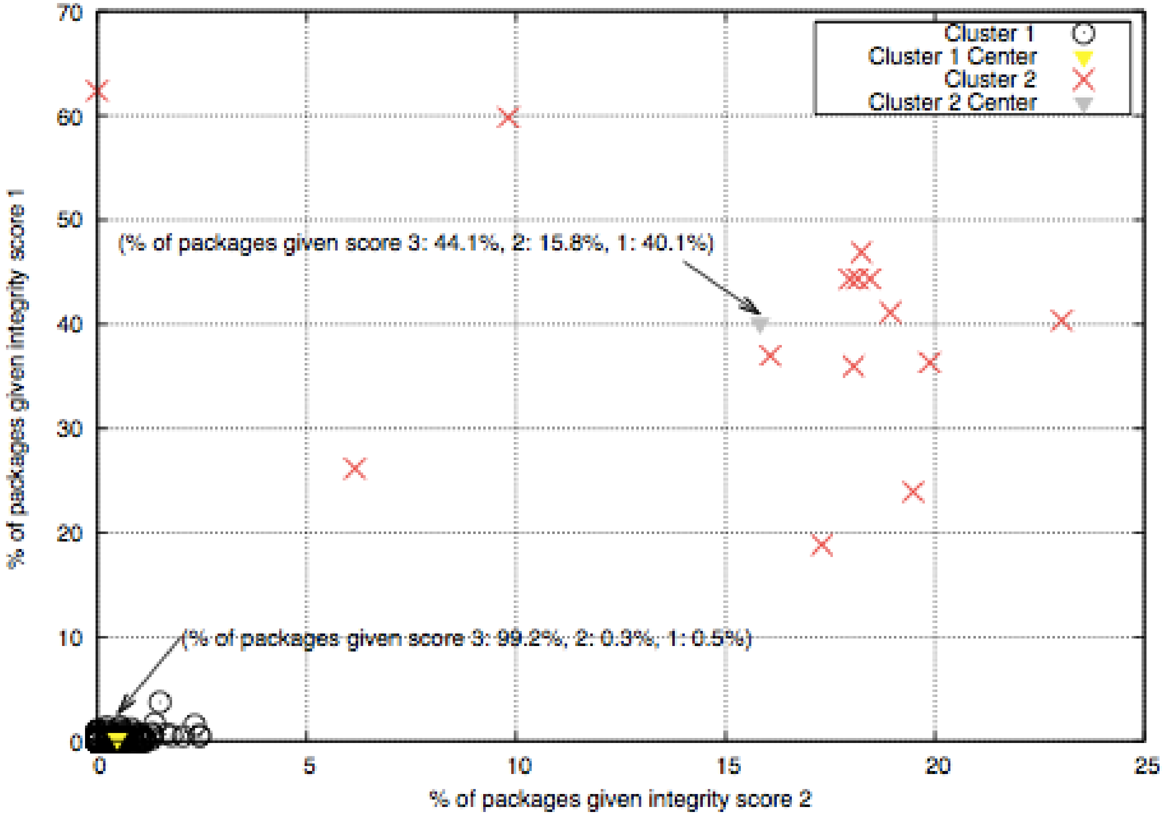


Usefulness of the framework

- Study software integrity of real-world VAs
  - Assessed through a software *whitelist*-based framework
  - Evaluated 151 Amazon VAs
- Our study shows significant variation in software integrity
- We demonstrate
  - The *need* for a whitelist-based framework to verify VAs
  - *Feasibility* and *scalability* of using whitelists for integrity assessment

Classification based on the outliers

- Formed natural clusters based on % of packages given scores 1 and 2.
- *k*-means clustering was used to identify two clusters, *k*=2



Characteristics of the low-integrity VA cluster

- Taking a closer look at 14 potentially untrusted VAs,
  - Significant portion of unverified files is common system files like /bin/cut and /bin/grep
  - 14 VMs are from different publishers/sources, were built to provide different functions

- Virus scanners flagged only 7 of the 14 VAs as malicious
  - What about the other 7?
  - In total, 41 of 111,981 unverified files were infected

None of them mentioned anything about software customization efforts!

Usefulness of the framework

- Our findings demonstrate the need for *a priori* software integrity assessment of VAs.

Table 7: Example VAs that provide php 5.3

		All files	*Unverified/ missing files*	*# software*	③	②	①
✓	*VA 1*	34,410	21	351	351	0	0
✓	*VA 2*	28,606	27	352	352	0	0
	VA 3	63,719	48	472	470	2	0
	VA 4	90,977	94	711	707	3	1
✗	*VA 5*	101,582	617	711	454	123	134
	VA 6	91,873	886	715	711	4	0
✗	*VA 7*	74,039	2,018	641	301	103	237

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- **David Nicol - Quantifying Trust in the Cloud**
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Motivation & Problem

- Hadoop widely deployed in cloud computing to handle big data (soon YARN)
- Hadoop focuses on only performance
- "Trust in application has multiple attributes, e.g.
  - Latency
  - Throughput
  - Reliability
  - Integrity
  - Confidentiality
  - Privacy
- Problem: Develop methodology for estimating trust attributes for a given context, as a function of observed system behavior that may reflect presence of intrusion / insider misbehavior

Approach

- Use Möbius tool to estimate trust attributes based on compositional model of
  - Application structure, workflow
  - Cloud middleware
  - Cloud resources
  - Presence / impact of cyber attack
  - "Belief" of system in different attacks, given observations
- We model uncertain relations among relevant elements of the cloud and the uncertainty of autonomous agents, by using Bayesian estimation, belief degree interval, and simulation.

Synergies

- Estimate run-time security attributes (Collaboration with Ravi/Zbigniew's group, and Bill's group)
  - Assume: a cloud system (Data center) has IDS to generate security alerts
  - Given observed alerts or identified compromised users, infer the degrees of belief in downgraded services in the cluster.

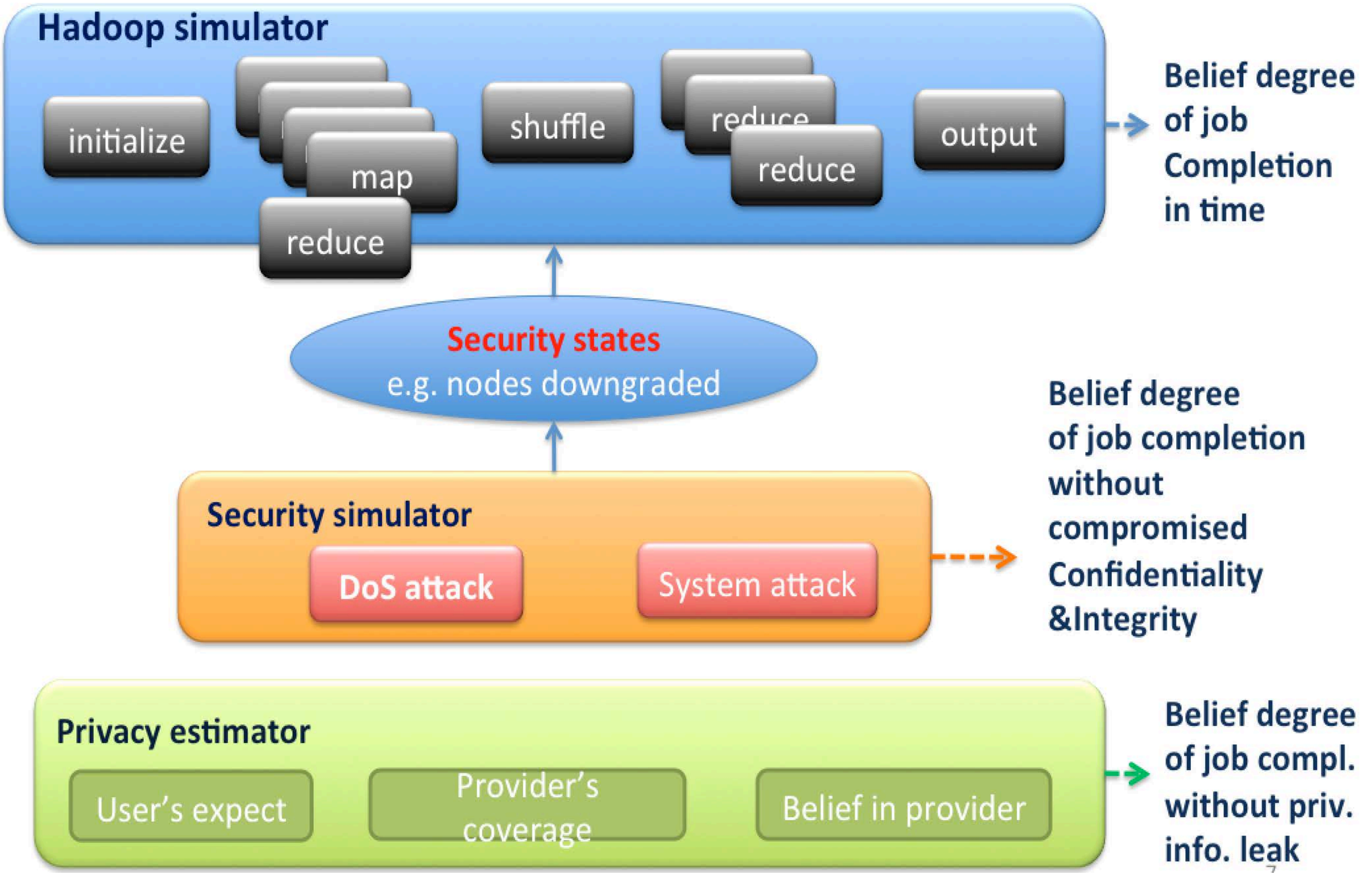
- Estimate performance attributes (Collaboration with Roy's group and Indy's group)
  - Given attributes of a cluster of hosts and a YARN application (workflow), and security attributes above, calculate the degree of belief in the completion of the workflow within a time limit.

- Estimate privacy attributes (Collaboration with Masooda's group) [We will add in, in recent future]
  - Given a cloud provider's privacy policy (and ToS), and a user's expectation profile, calculate the degree of satisfaction on privacy protection

Demo Overview

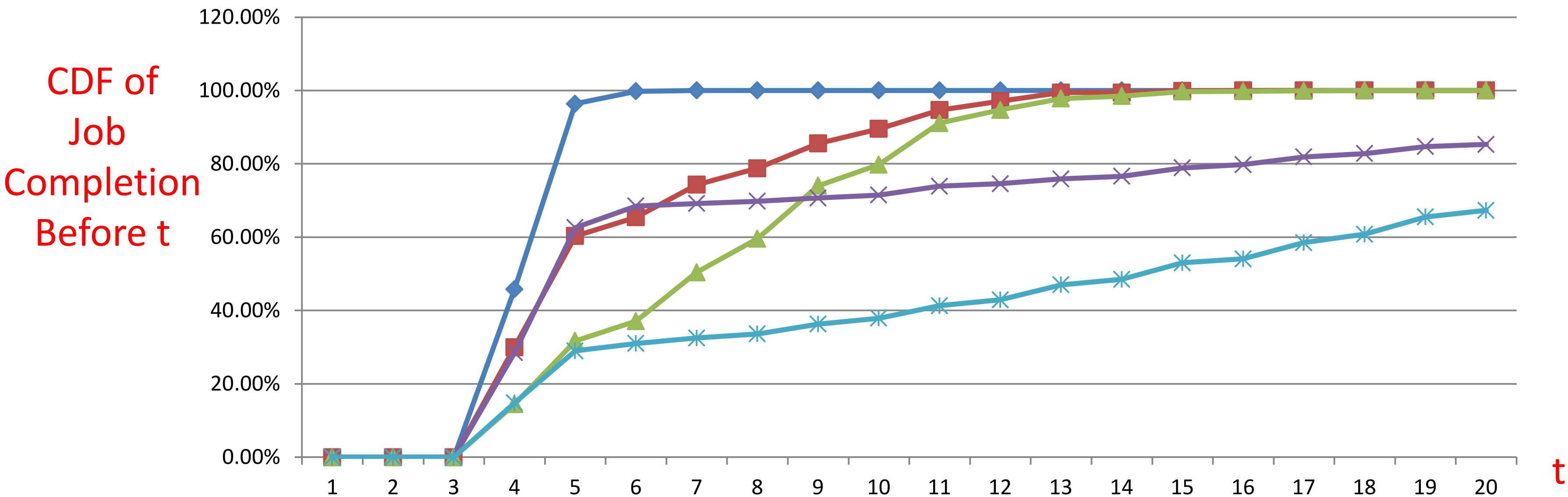
- Overview
  - A MapReduce job (it could be a more complex DAG of tasks) for a critical mission, running on Hadoop YARN system in a public or community cloud;
    - 128 mappers
    - 64 reducers
    - 1000 nodes of cluster
    - The YARN system uses Capacity Scheduler.
- Example threat
  - Current YARN system considers only memory, to represent the computing resources sharing among cloud users, without considering CPU, disk read/write, and network bandwidth;
  - Attackers and/or malicious users can launch DoS attack by exhausting computing resources of the hosts running their "tasks".

Demo Overview



Scenarios

- S1: no security attacks, with capacity: 5%;
  at 6, the degree of belief in job done is 99.8%
- S2: "mild" (50% of CPU) DoS attack by a single user with capacity: 0.5%
- **S3: strong (90% of CPU) DoS attack by a single attack (capacity: 0.5%)**
- S4: "mild" (50% CPU)DoS attack by a single user with capacity: 5%
- S5: strong (90% CPU) DoS attack by a single user with capacity of 5%; job cannot complete after 30 time units (5 times longer than normal)



Results

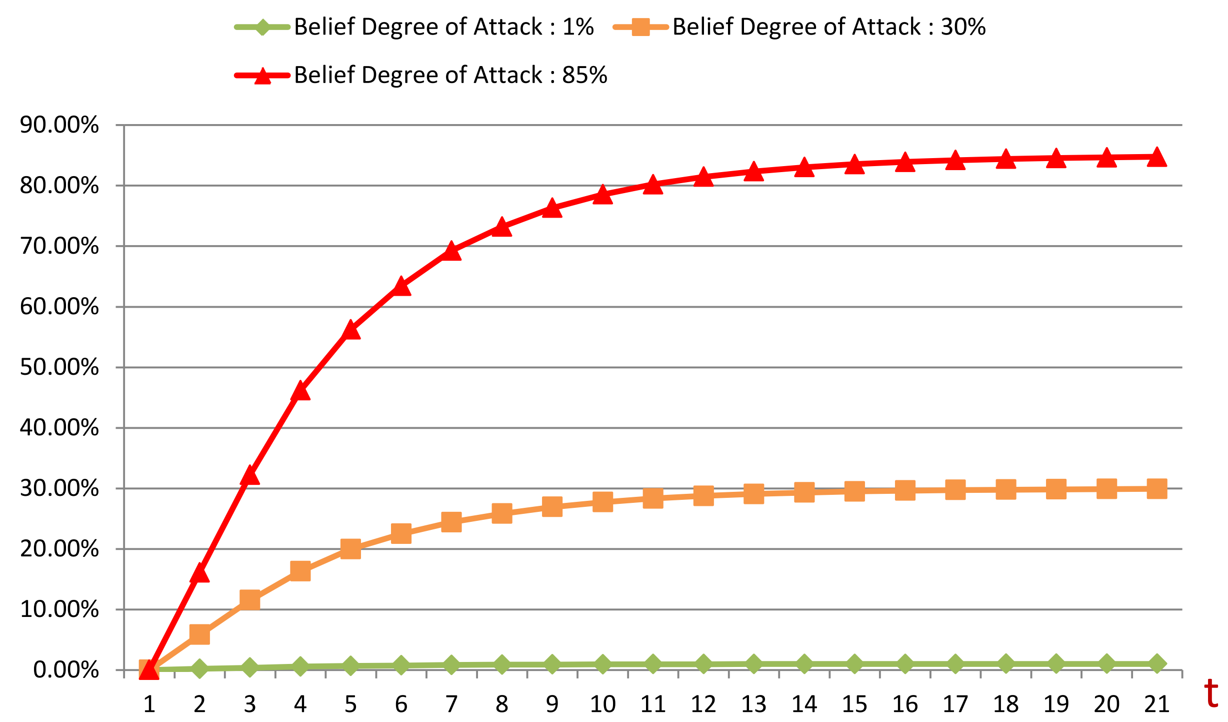
- Powerful modeling tool allows expression of different components of system (application, middleware, hardware, attack/response) to estimate coupled trust attributes

- Finishing time of a given job is sensitive to relatively few number of **legitimate** but **resource-intensive** tasks
  - Intruder stealing credentials or insider can upset balance

Confidentiality and Integrity

- In the cases that the nodes in the cluster compromised,
  - Can compromise confidentiality and integrity.
- Belief Degree of Integrity Compromising (similarly, Confidentiality)

CDF of Integrity Compromising Over time



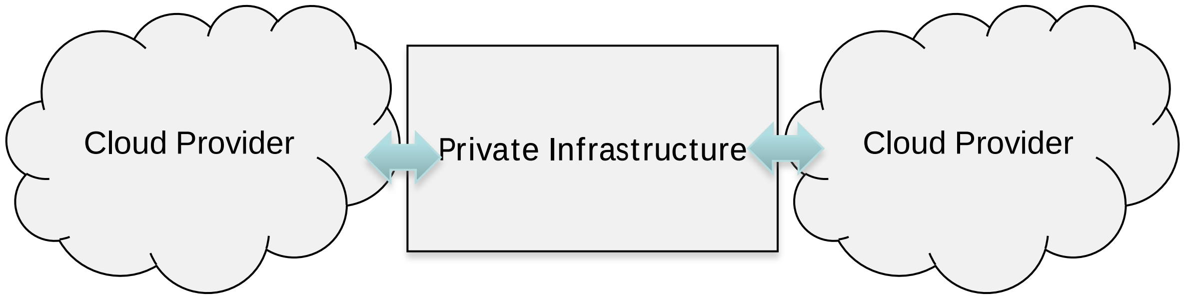
Future Work

- To model YARN with Möbius with higher resolution, as a basis to support innovative system design and validating;
- To improve YARN schedulers for optimizing YARN application (workflow), by considering security and privacy attributes;
- To work on cross-cloud workflow trustworthiness estimation and optimization;
- To further formally characterize Hadoop/YARN system's trustworthiness.

Research Results at Center

- **Indy Gupta - Real Time Challenges in Clouds**
- **Jose Meseguer - Model-checking NoSQL Storage Systems**
- **Rakesh Bobba - Group Key Management as a Service**
- **Jose Meseguer - VA Admission Control**
- **David Nicol - Quantifying Trust in the Cloud**
- **Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance**
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Motivation: Multi-Domain Monitoring



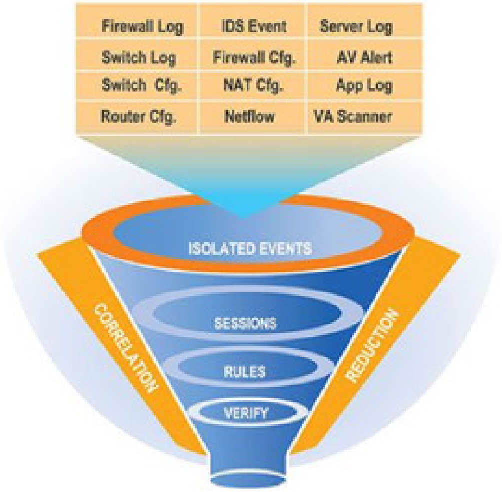
- Modern systems are composed of multiple security domains
  - Cloud Computing
  - Hybrid Clouds
  - Intercloud-Multi-cloud
  - Critical Infrastructure Systems
- Advantages
  - Economy of scale for cloud computing
  - Ability to select which services to use without binding to a single provider for multi-cloud/inter-cloud

EVENT TYPE	SOURCE	SECURITY DOMAIN	DESCRIPTION
runsCriticalService	Deployment software,SNMP agents	Cloud user	critical services run on a specific instances
instanceAssigned	Openstack	Cloud provider	instances are assigned to specific physical servers
badTraffic	IDS, Network monitoring	Cloud provider	malicious traffic detected from specific physical server

Problem: Sharing Only need-to-know Information

- How do we know that the system is working correctly as a whole?
- Integrating events across domains to detect complex security problems and attacks
  - **S**ecurity **I**nformation and **E**vent **M**anagement Systems (SIEM) are successful because they are capable of integrate monitoring events across multiple sources

(SIEM)



- However, monitoring provides **critical information** about systems to external organizations which opens the system to attacks
  - *Network topology, network traffic, configurations, installed programs, vulnerable programs, user bahaviors, services, critical machines*

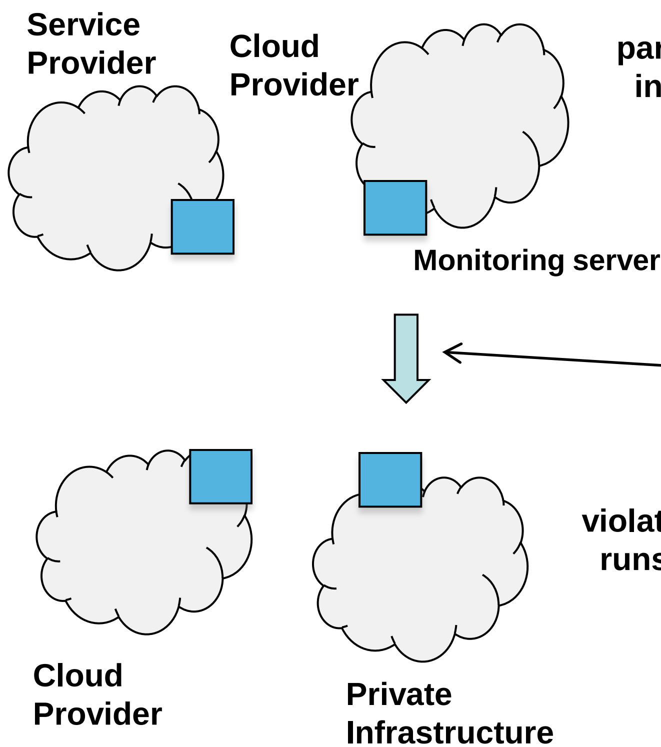
Goal: Share only ***need-to-know*** information across organizations to detect problems

Approach: Policy-based Distributed Monitoring System

- Our Distributed Monitoring System
  - Policy-based
  - Detects violations of global policies while limiting event exposure
  - Identifies "need-to-know" events and shares only those

1. Policy rewritten to identify cross-domain sharing
2. Events shared only if they can create a violation

violation(I, P) ← runsCritService (I, P), instAssigned(I, S), badTraffic(S)



partial(I) ← instAssigned(I, S), badTraffic(S)

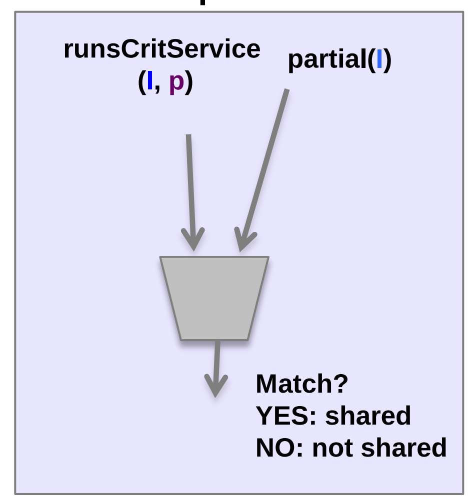
Processed by the cloud provider when information about physical servers are received

partial(i) is shared only if the rule on the receiving side can be satisfied

violation(I, P) ← runsCritService (I, P), partial(I).

Processed by the cloud user after receiving partial(I) from the cloud provider

Secure Two-Party Computation



Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- **Roy Campbell -** Monitoring, **Failure Scenario as a Service**, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Motivation

- Consequences of moving to cloud and using commodity hardware
  - Increase in system's dependability
  - Increase in equipment failure

REQUIREMENT #1:
Cloud applications normal operating procedures should be prepared for different failure scenarios.

- What happens when failure occur?
  - Increase in maintenance costs
  - Increase in applications down-time could cause them to:
    - Miss deadlines
    - Be non-compliant with Service Level Agreements (SLA)

REQUIREMENT #2:
Identify resources, the failure of which could jeopardize missions

Problem: What to do when failures occur?

- What can we do to deal with failures?
  - Inject failures→ using Chaos Monkey (Netflix) or Anarchy Ape (Yahoo!)
  - Observe critical resources
  - Increase resource availability by:
    - Using more robust hardware
    - Increasing redundancy
- Contributions:
  - We classify Hadoop-type applications based on different types of intensive workloads including:
    - ❑ I/O – random writer
    - ❑ CPU – word count
    - ❑ Network – text sort
  - We enable users to chose configurations of Hadoop- type applications to better survive failure scenarios

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- **Roy Campbell -** Monitoring, Failure Scenario as a Service, **Data Assurance**
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Motivation

- Big Data shifts bottleneck from computation to I/O
- Storage layer can suffer from:
  - Hotspots
  - Reduced performance
- Storage layer problems can affect jobs operating on the data

Reduced performance → Missed deadlines → Mission failure

- Adequate design and performance tuning of storage layer is <u>critical</u>

Problem

- Design and tuning must be validated by specific workloads
  - Observed workloads
  - Predicted workloads

- Incorrect workload assumptions ➔ sub-optimal performance
  - Increasing costs
  - Leading to job delays (and ultimately mission failure)

Approach

- Model workload using a combination of:
  - Statistical measurements
    - Empirical distributions from data
  - Delayed renewal processes
    - Model temporal locality: short-term temporal correlations and file popularity
  - Unsupervised statistical clustering
    - Use k-means to find objects with similar behavior

    ➔ Type-aware workload modeling

Motivation

- Model for efficient replica computation for improved availability of files based on priority classes

Files are treated all the same in today's distributed file systems(HDFS, GFS). How can we prioritize the availability of files essential to the success of the mission?

Mission Data	Business Data	Development Data	Research Data

Problem

Given a certain amount of storage space, how should we distribute the storage space to different file classes with different importance weights, and achieve the highest overall weighted availability?

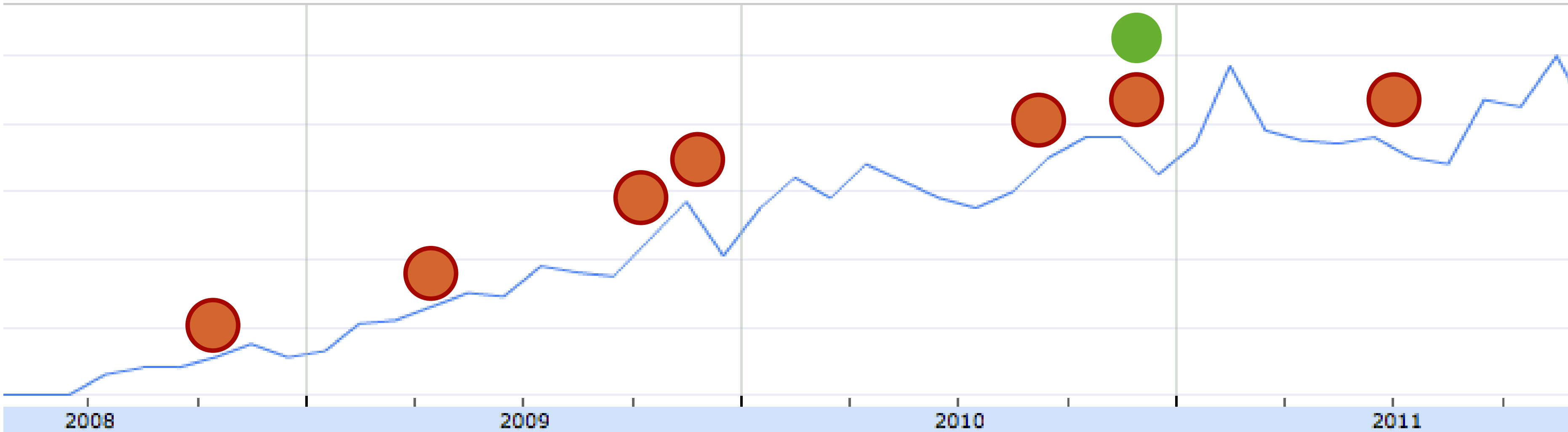
Approach

Since the original optimization problem is NP hard, we propose a greedy algorithm called Class and Budget(CaB) which always assigns storage space to the file class which can improve the overall availability the most. Compared to an algorithm which assigns storage naively proportional to file weights, CaB utilizes storage spaces more efficiently since it does not create unnecessary replicas.

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- **Ravi Iyer - Monitoring Driven Trust**: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Clouds: Security Problems



Jul'08 - Spammers set up mail spamming instances in the Amazon's EC2 cloud.

Sep'10 - Google Engineer Stalked Teens, Spied on Chats

Apr'09 - Texas datacenters operations are suspended for FBI investigation.

Dec'10 - Microsoft BPOS cloud service hit with data breach

Nov'09 - Side channel attack of Amazon's EC2 service.

June'11 - Dropbox: Authentication Bug Left Cloud Storage Accounts Wide Open

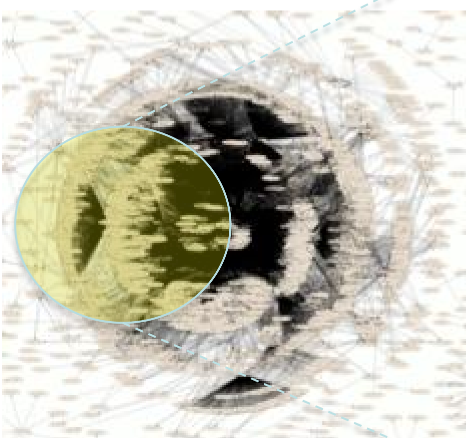
Dec'09 - Zeus crime-ware using Amazon's EC2 as command and control server.

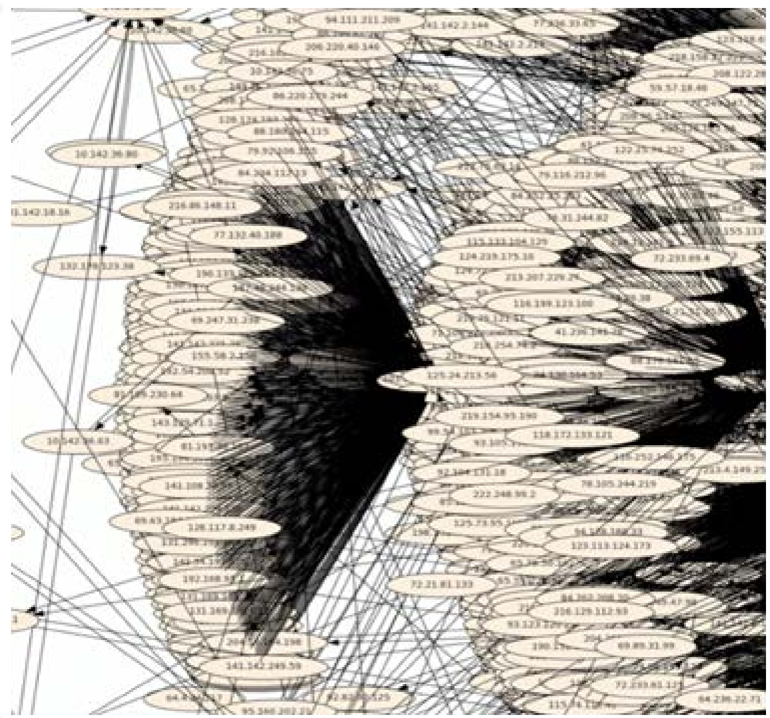
Dec'10 - Anonymous hacker group failed to take down Amazon

Analysis of Security Attacks from a Large System: NCSA Case Study

- Goals:
  - Provide the system-level characterization of incidents and evaluate the intricacies of carrying out successful attacks
  - Measure the efficiency of the detection and diagnostic methods

- Challenges

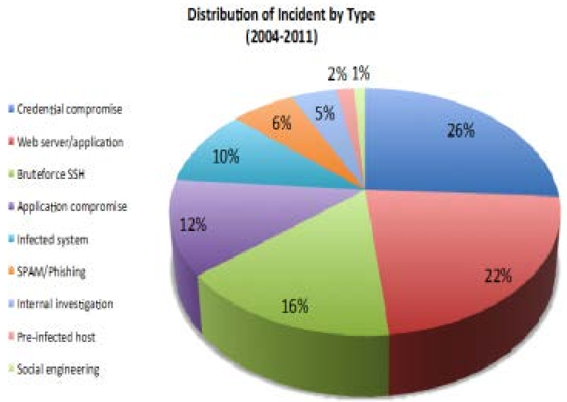




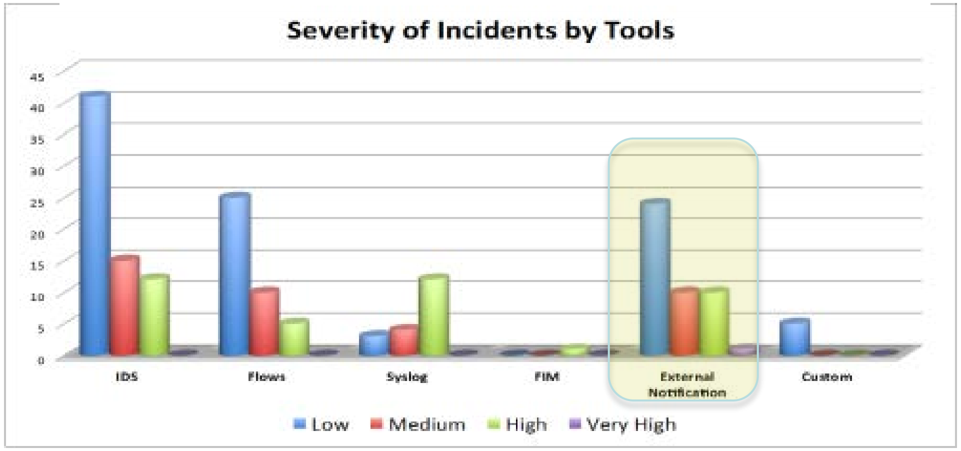
Five-Minute Snapshot of In-and-Out Traffic within NCSA

Sample Real attack Data: Missed Incidents

Distribution of Incidents by Type 2004-2011



Severity of Incidents by Monitoring Tools 2004-2011



Cause of missed incidents	Examples	#
Increased sophistication in attacks	A peer site gets compromised and attacker logs-in with stolen credentials; zero-day exploits	6
Lack of signatures	Exploit of VNC null string authentication vulnerability	7
Admin misconfiguration	Web share world writable access or root login to accept any password	5
Inability to distinguish traffic anomalies in the network	Web defacement or use of web server to host malware; bot command and control traffic	10
Misconfiguration of security monitoring tools	Routers stop exporting the flows to central collector which prevents alerting	1
Inability to distinguish true positives from false positives	Human error	2
Inability to run monitors on all hosts and file systems due to cost	Limited deployment of file integrity monitors on non-critical systems	3

significant portion of undetected incidents have high and very high impact (severity)

25% of incidents are missed (undetected)

Observations from Real Attacks

- **Need for *Continuous Monitoring* to pre-empt an attacker actions**
  - potentially let the attacker to progress under *probation* (or tight scrutiny) until the real intentions are clear
- **Need to correlate**:
  - data from different monitors
  - system logs
  - human expertise
- **Need to validate benchmark success against real data**

Execution Under Probation

Real-time Analytics

Preempt attacks

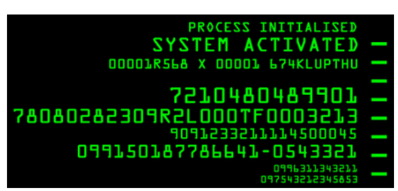
Conclude and block attacks

Attack Prediction

Continuous Monitoring

Probation Environment

Learn attackers' behaviors



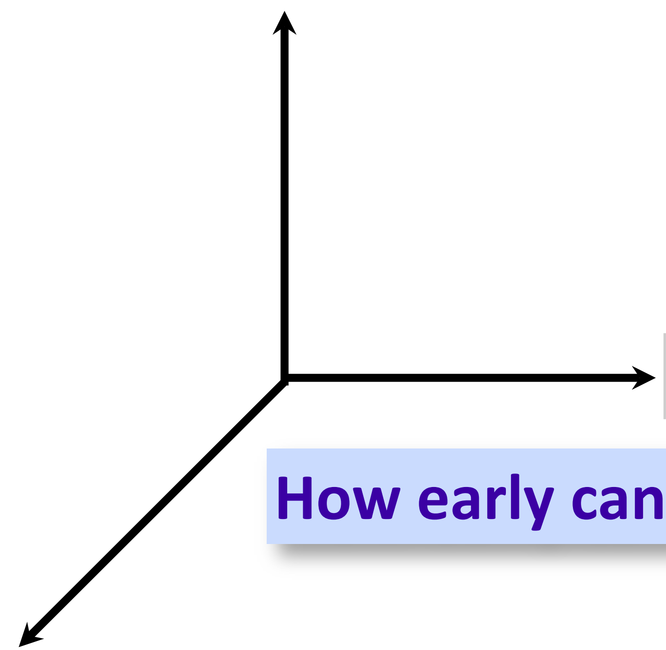
operational data

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- **Ravi Iyer** - Monitoring Driven Trust: **Execution under Probation**
- **Zbigniew Kalbarczyk -** Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Fundamental Tradeoffs

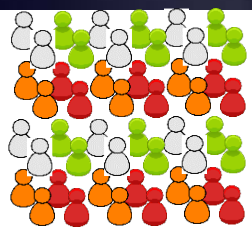
How much does monitoring cost?



How early can we identify attackers?

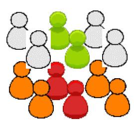
What is the desired detection accuracy?

Execution Under Probation Effectiveness



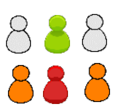
1021 users + **alerts**

Compute Suspicion Score



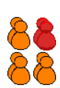
232 suspicious users

Monitor in Probation Environment



42 had more than 3 alerts

Block Suspicious Activities



14 attackers of a total 15 attackers

1. Compute Suspicion Score using:
Past: use ground truth data to compute likelihood
Present: use alert disorder, alert rate, and decay factor
2. Select top suspicious users

1. Look for users that generate more than three alerts in probation environment. They are attackers.
2. Return other users to normal execution environment.

Block suspicious commands, e.g., "sudo" to prevent privilege escalation. We use a learned dictionary of suspicious commands.

That means 90+% detection rate. We miss 6.67% of attacks - considerably better than 27% misdetection rate of previous study (Sharma et. al., DSN 2011)

Execution Under Probation

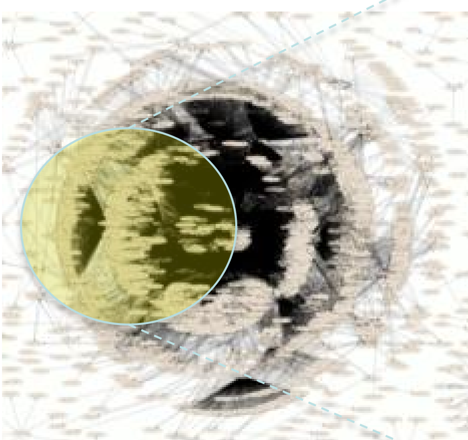
Challenges Going Forward

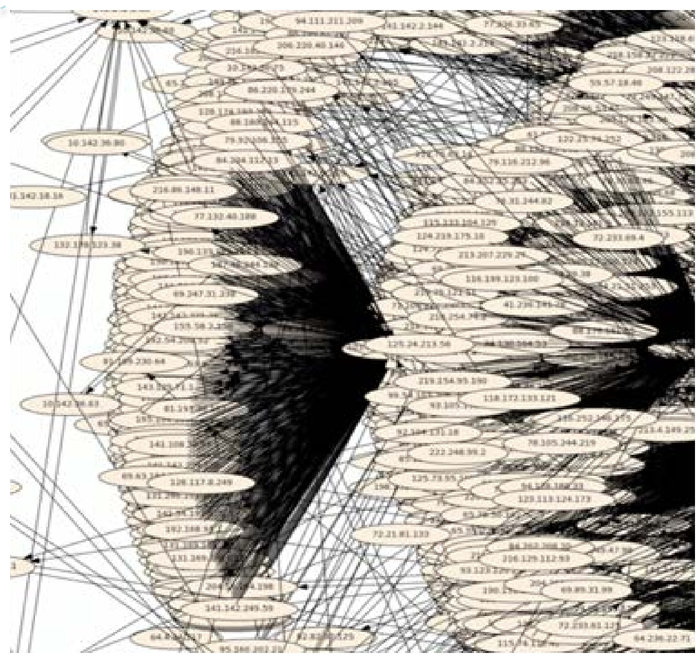
- Developing orthogonal views by monitoring at different levels and granularities -to fully understand it’s underlying attack/user actions.
- Robust Methods to pre-empt malicious actions in a timely manner
- Evaluate effectiveness against classes of attacks

Monitoring and Invariance Driven Trust

- Goals:
  - Design attack independent protection strategies: minimize number of missed incidents and false positives
  - Invariance driven trust
  - Demonstrate techniques in an experimental testbed

- Challenges



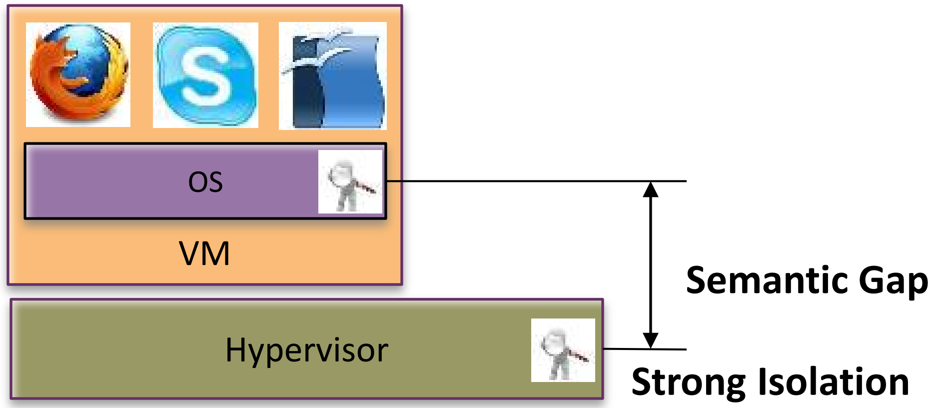


Five-Minute Snapshot of In-and-Out Traffic within NCSA

Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- **Zbigniew Kalbarczyk - Building Resilient Virtual Machines**
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- Gul Agha - Coordination and Probabilistic Consistency

Challenges in VM Monitoring



- **Challenge: Semantic Gap**
- **Curr**
  - U ... virtual
    r
  - Limitations:
    - Require effort to understand the guest OS
    - Monitoring tools need to be updated as the guest OS updated
    - Share the same view with attacker: can be manipulated

Our Solutions:

VM Introspection based on

the **Architectural Invariants** of VM environment

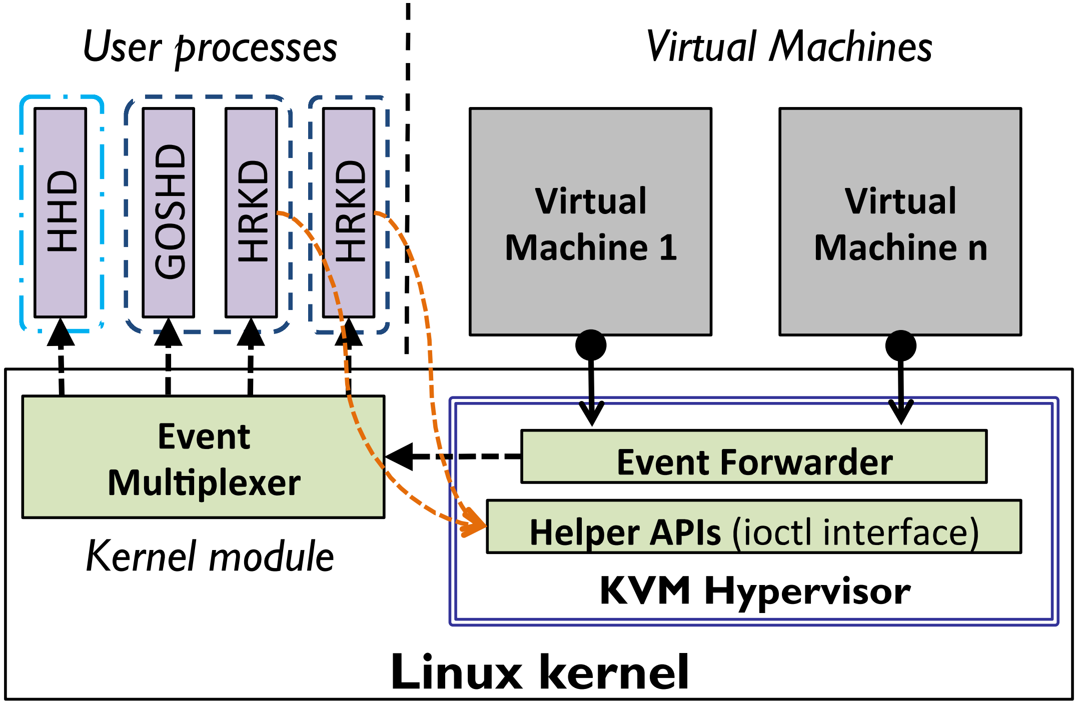
What Do We Monitor?

- Guest system's architectural state
  - VM Events, General Purpose and Control Registers
- Guest system's virtual devices
  - Network interfaces, hard disks, memory
- **Advantages:**
  - Non-intrusive to the guest system
  - Hypervisor independent
  - Guest system independent
  - Detection based on the system/application behavior

Reliability and Security Checkers

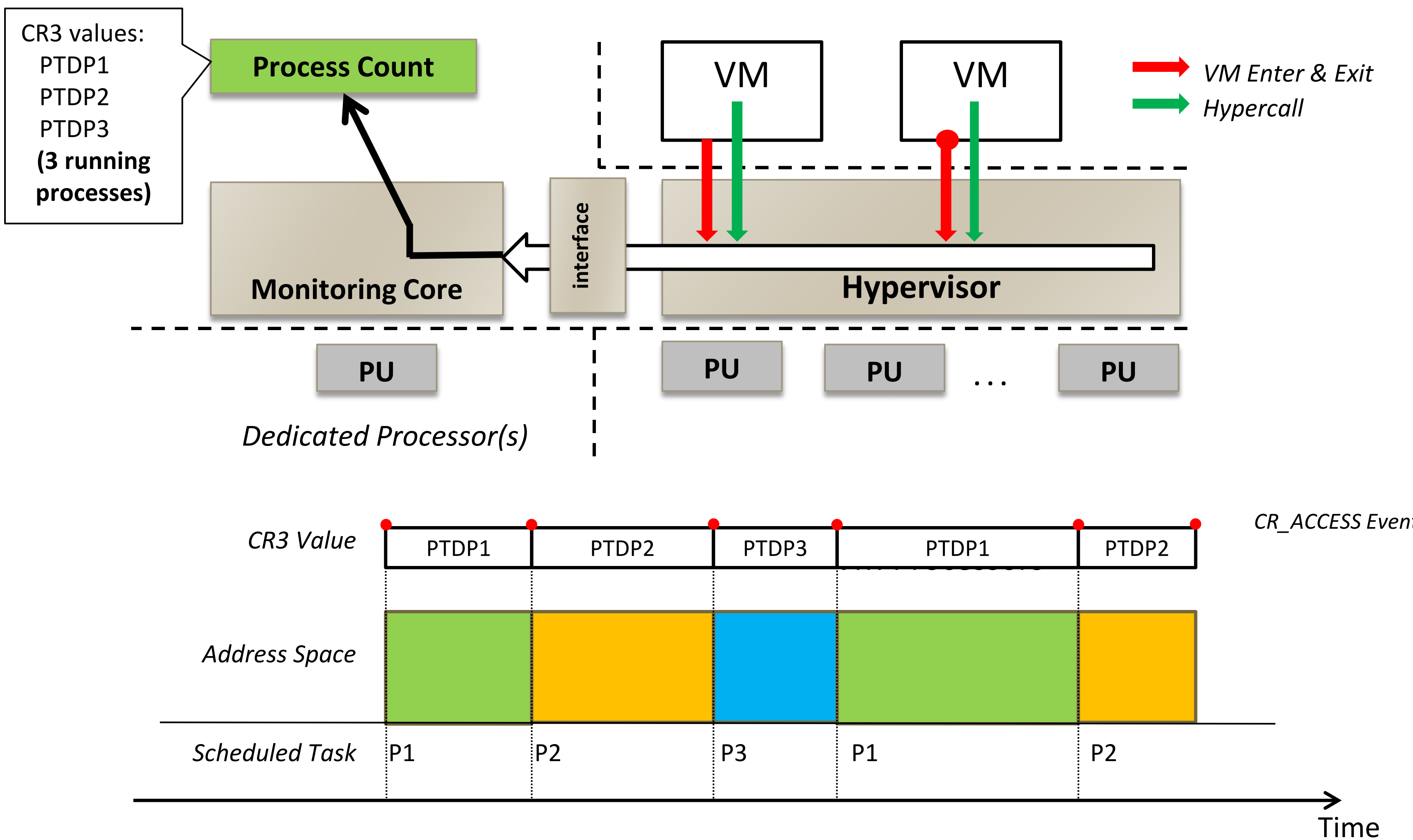
- **Guest OS hang detection**
  - Infinitive time between two consecutive context switches
- **Hidden rootkit process detection**
  - The number of running processes displayed by the guest system (Task Manager, PS, TOP) is smaller than the number reported by our monitoring tool
- Hypervisor hang detection
  - Infinitive time between VM Exit and VM Entry events
- Guest OS boot sequence integrity
- Process termination detection
- ….

Framework Integration with KVM Architecture



- Example detection modules:
  - Hypervisor hang detection (HHD)
  - Guest OS hang detection (GOSHD)
  - Hidden Rootkit detection (HRKD)

Hidden Rootkit Process Detection



Future Work: On-line Attack Containment and Mitigation

- Employ the proposed framework as a basis for preemptive (i.e., before a system misuse) detection of compromised user and mitigation of attacks
  - Explore new monitoring/detection techniques to contain an attacker and hence, prevent an attack propagation
- Consider vulnerabilities and malicious attacks against hypervisors
  - Hardening hypervisor against potential attacks
- Develop methods and tools to experimentally assess the proposed solutions
- Demonstrate the proposed approach in cloud environment while running representative applications on our testbed

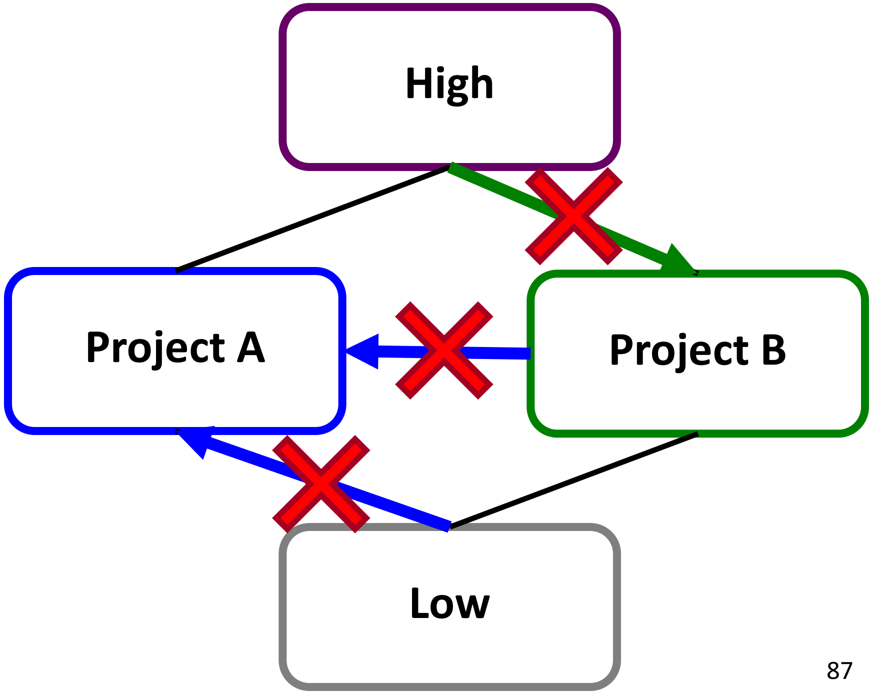
Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- **Bill Sanders - Hardening the Cloud with User Account Monitoring**
- Gul Agha - Coordination and Probabilistic Consistency

User Account Monitoring and the Cloud

- If compartmentalized projects use a common, general-purpose cloud infrastructure, then access-control violations may occur

- Consider Bell-LaPadula policy:
  - Mandatory Access Control (MAC)
  - To ensure confidentiality:
    - No write down
    - No read up
    - No lateral reads/writes



Background

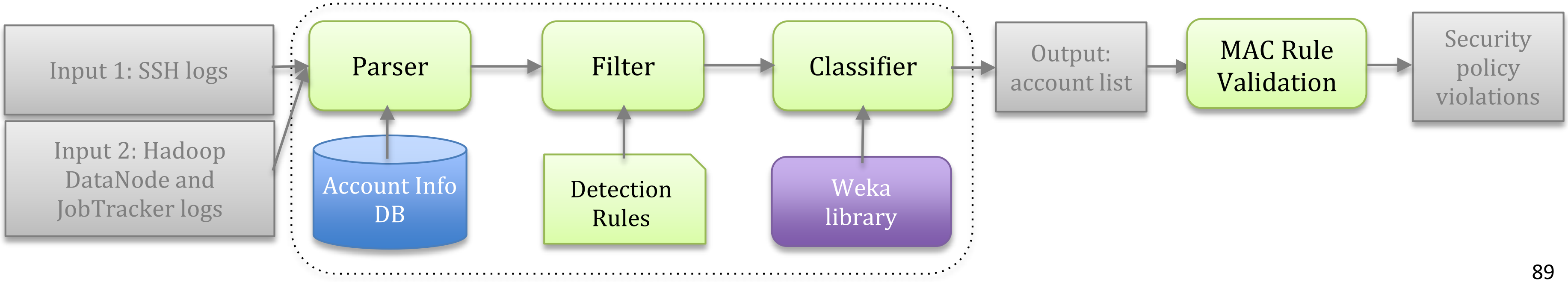
- In previous work, our University Credential Abuse Auditing System (UCAAS) successfully detected and flagged compromised accounts
  – Collaborated with CITES and the University of Michigan
  – Used real VPN and Bluestem logs (45,000+ users)
  – Delivered to the Air Force

- We extend UCAAS to monitor Hadoop clusters for security policy violations in a private cloud setting

Technical Details – UCAAS Architecture

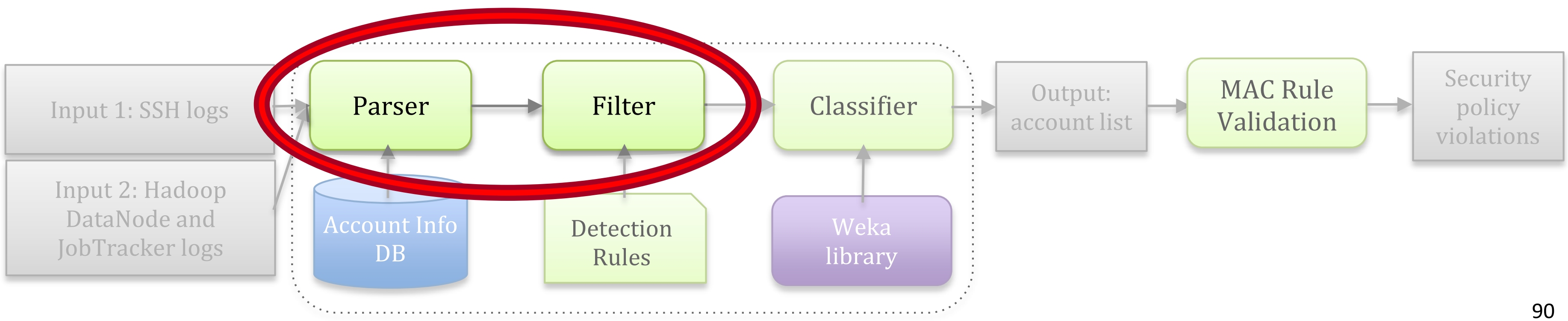
- We use UCAAS to detect compromised accounts within a Hadoop cluster on an organization's cloud

- We apply security policy rule validation on the logs for the compromised account to automatically determine lattice policy violations



Technical Details - Features

- Some suspicious behavior features used to detect account compromise:
- IP addresses accessing multiple accounts
- Large number of failed SSH attempts
- Access of others' Hadoop machines
- Access of other users' DataNodes
- Large number of failed MapReduce job attempts



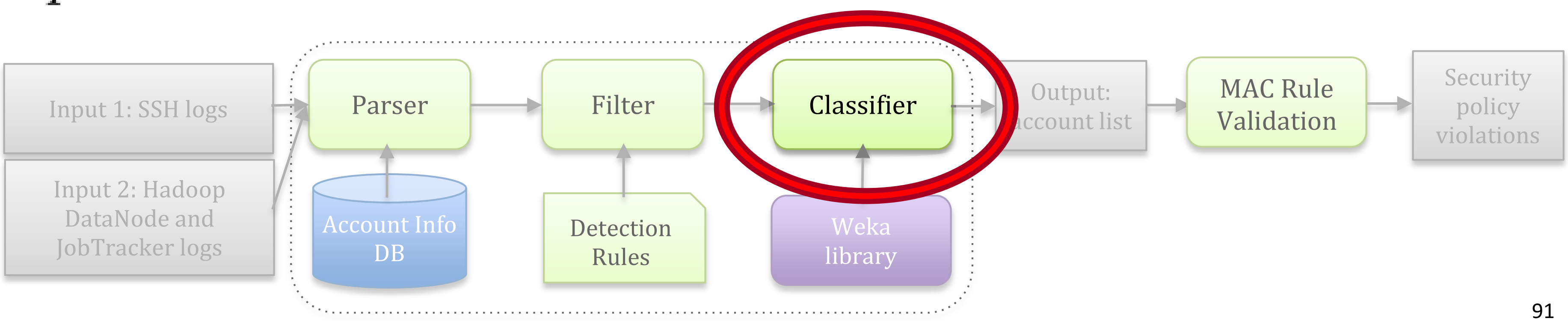
Technical Details – Machine Learning

- UCAAS uses machine learning to determine suspicious activity
- Logistic regression classifier written in Weka

- Dependent variable is Boolean (is there a compromise?)
- Estimated regression model: $L = a + \sum B_i X_i$
- L is the natural logarithm of the odds that the events represented by the dependent variable happens

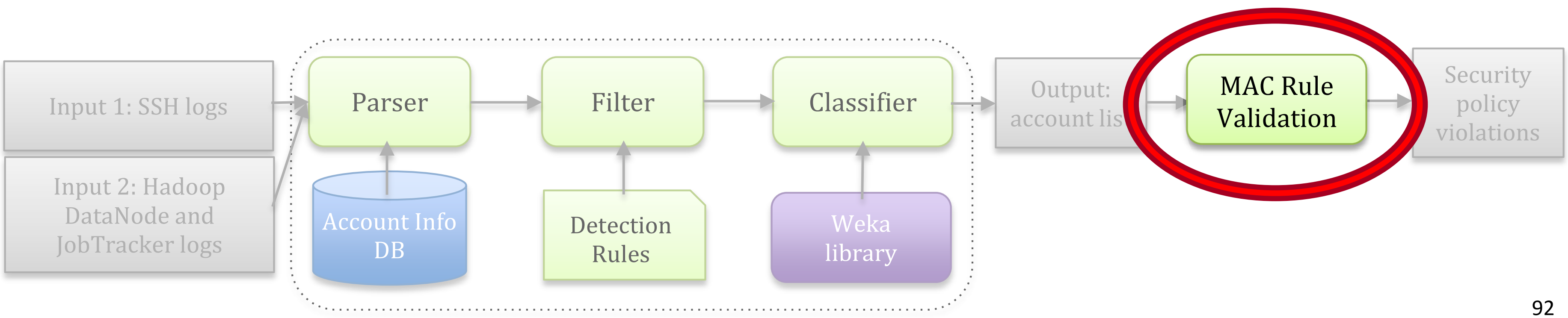
$$L = \ln \frac{\hat{p}}{1 - \hat{p}}$$

- $\hat{p}$ is the probability that the characteristic variable is true



Technical Details – Policy Validation

- We apply security policy rule validation on the extracted features for the suspicious accounts
- Rules are specified in a format that allows for execution against the extracted features
- Output is the security policy violations (if any) for the suspicious accounts



Conclusions

- UCAAS has successfully been used to detect compromise for university VPN accounts.

- We demonstrated UCAAS' use as a tool to enforce security policy for private clouds
  - The cloud's resource-abstraction may allow for violation of traditional Mandatory Access Control policies (e.g., Bell-LaPadula)
  - UCAAS lets us detect account compromise using logs available in the cloud
  - We can detect violations of security policy by validating features for the compromised accounts against policy-derived rules.

Future Work

Our work improves upon static, rule-based IDS systems

- Can learn about evolving threats
- Provides alerts relative to security policies

Part of a larger project to develop an intrusion response and recovery engine

Goals:

- Detect intrusions in real-time relative to business objectives
- Calculate responses that ensure operation within safety bounds
- Allow administrators to make cost-driven response decisions

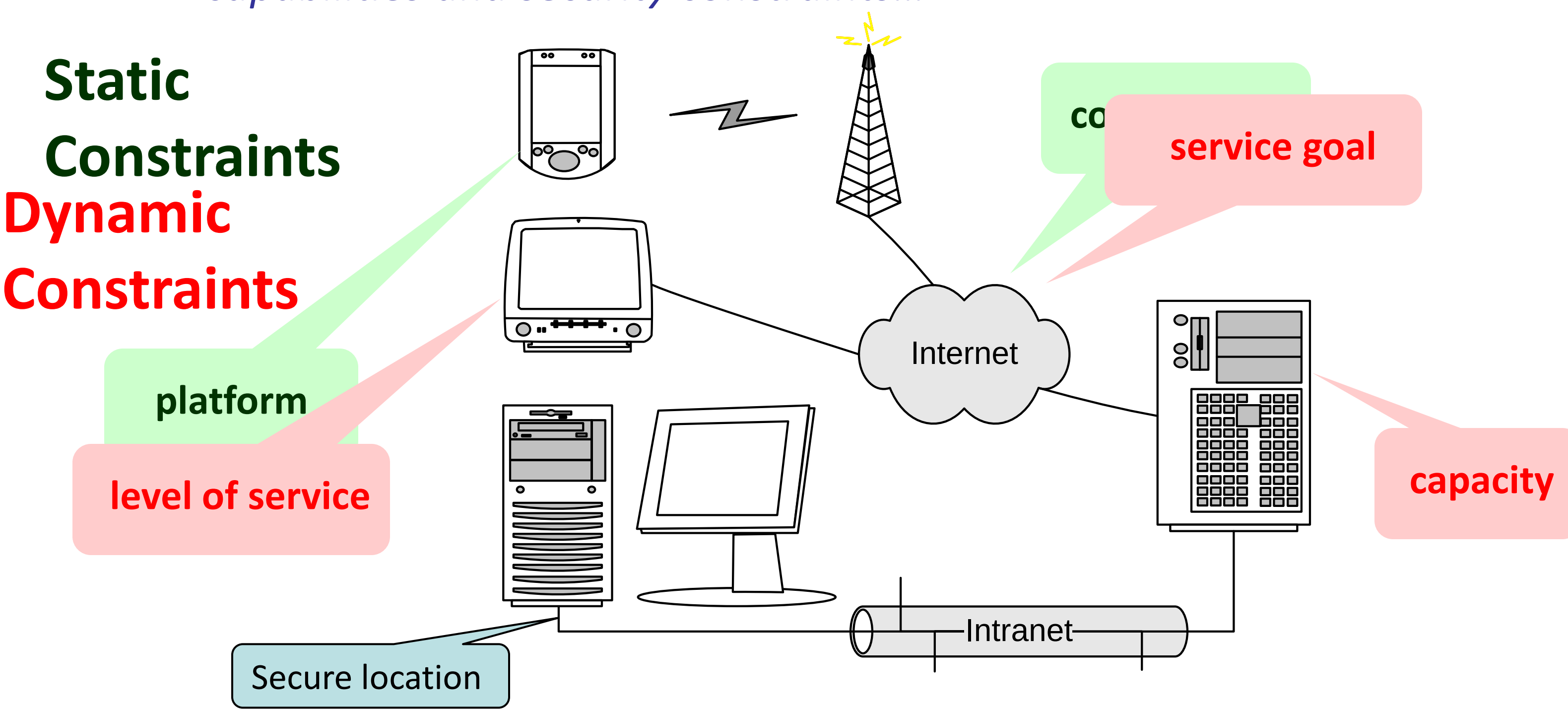
Research Results at Center

- Indy Gupta - Real Time Challenges in Clouds
- Jose Meseguer - Model-checking NoSQL Storage Systems
- Rakesh Bobba - Group Key Management as a Service
- Jose Meseguer - VA Admission Control
- David Nicol - Quantifying Trust in the Cloud
- Roy Campbell - Monitoring, Failure Scenario as a Service, Data Assurance
- Ravi Iyer - Monitoring Driven Trust: Execution under Probation
- Zbigniew Kalbarczyk - Building Resilient Virtual Machines
- Bill Sanders - Hardening the Cloud with User Account Monitoring
- **Gul Agha - Coordination and Probabilistic Consistency**

1. Motivation

Web Applications and Mobile Clouds

- *The Mobile Cloud is a collection of many platforms with different capabilities and security constraints…*



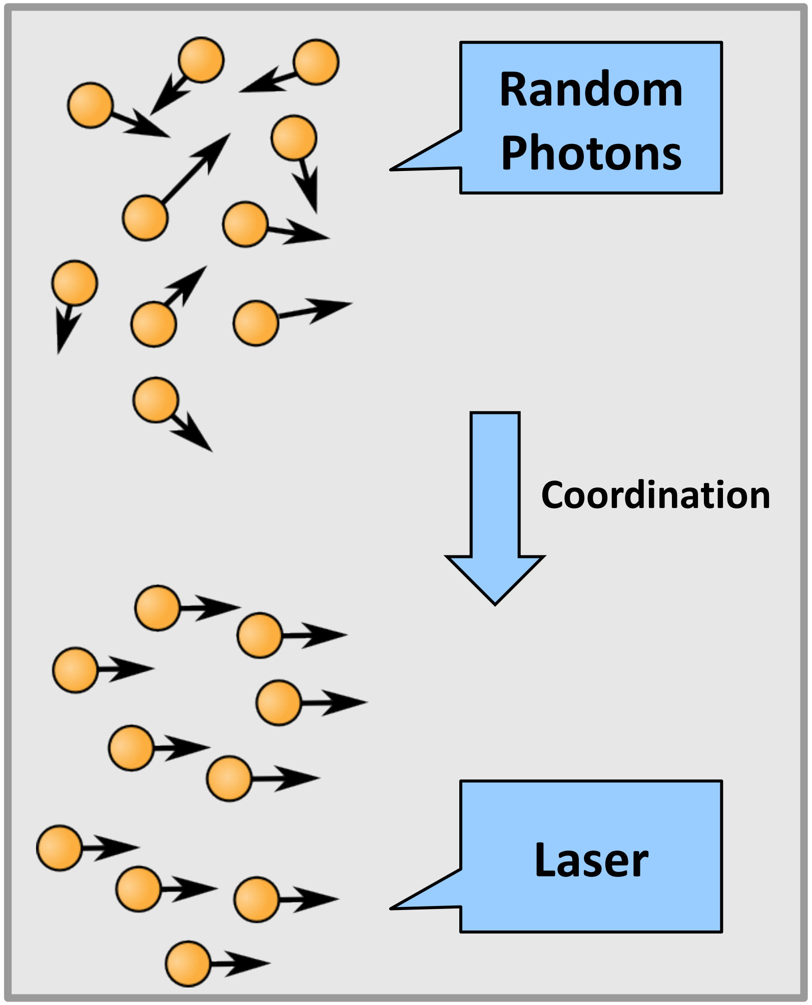
2. Problem

Coordination in Cloud Computing

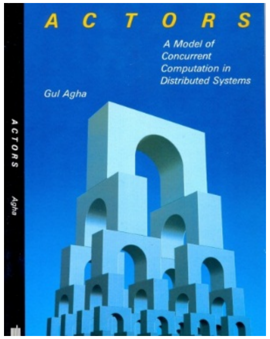
- The cloud provides:
  - Computation on nodes
  - Interaction between nodes

Completing a mission requires:

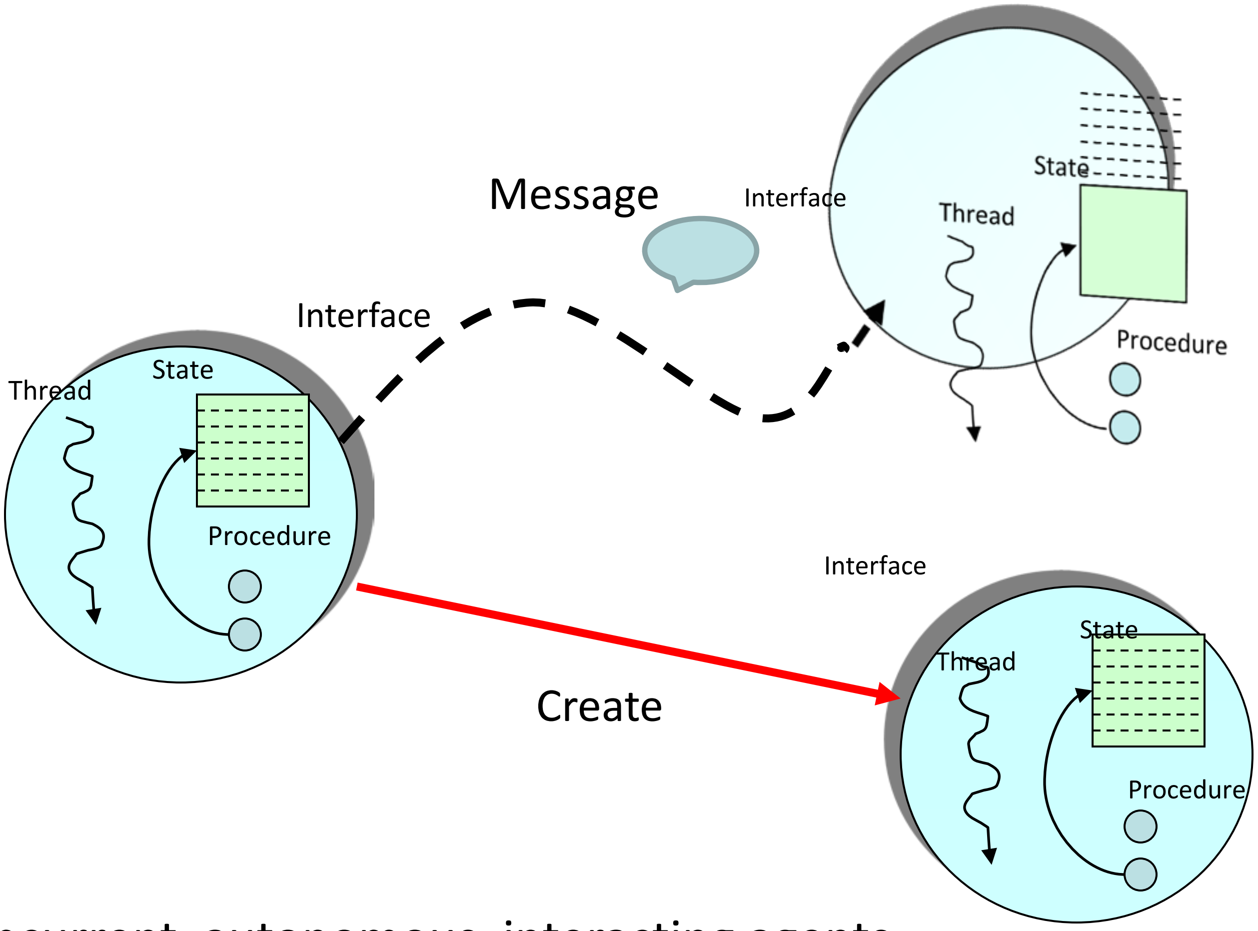
  - Computation on nodes
  - Secure and safe interaction of nodes



3. Approach



a cloud represents a collection of actors

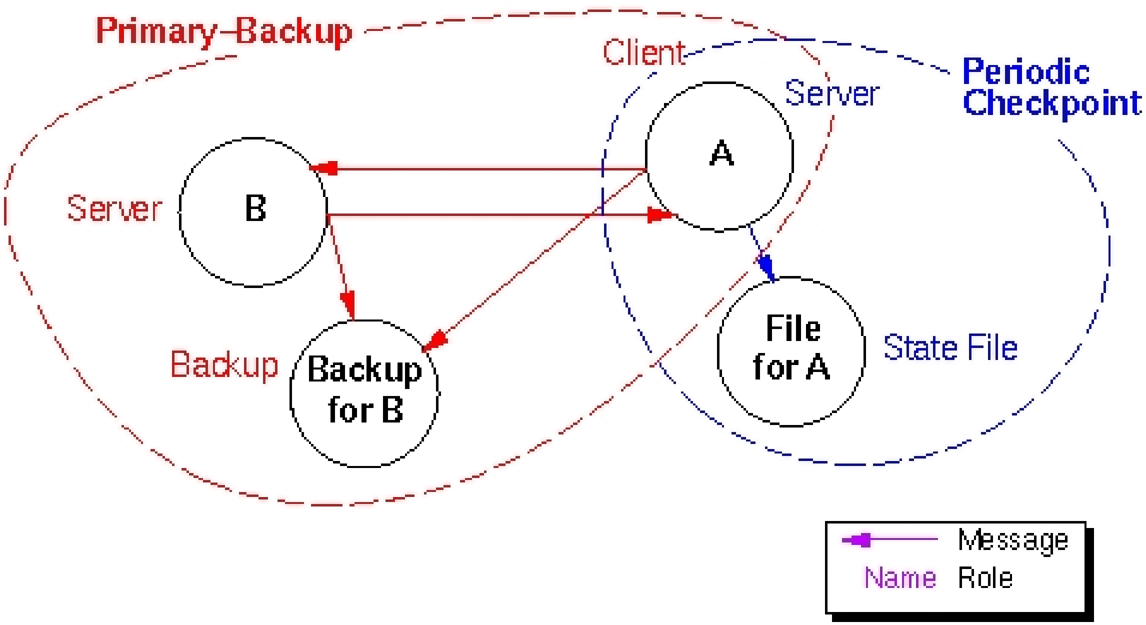


- Actors are concurrent, autonomous, interacting agents.
- New actors have their own address
- Addresses may be communicated in messages

The Actor Model

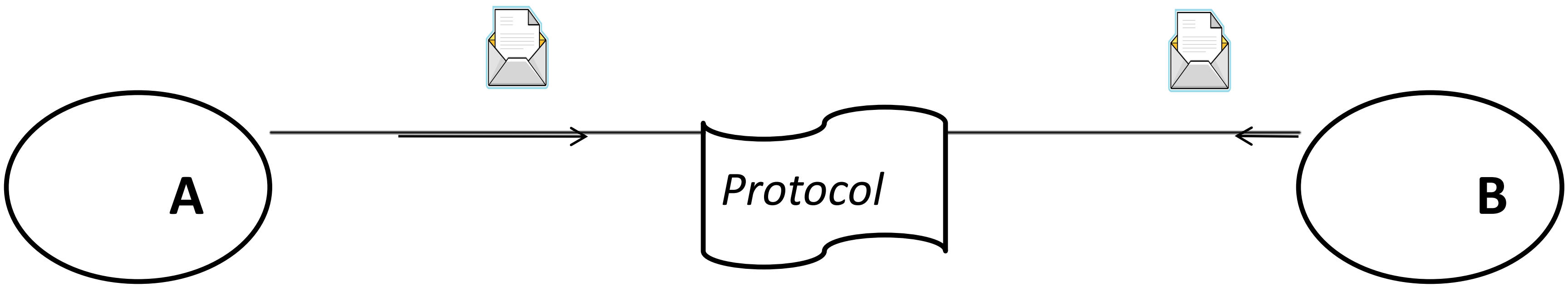
Abstracting Interaction between Groups of Actors

- A protocol defines a role for each participating actor relative to the protocol

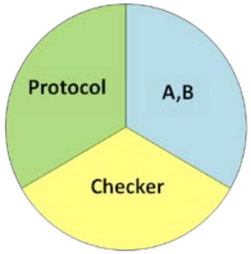


Coordination Protocols

- Specify contracts (protocols) to describe a session between a group of actors
  - *Session types*



- Statically and/or dynamically verify adherence to these contracts



Session Types

Global Types

The language in which protocols are specified.

The workflow

$$\left.\begin{array}{rcl} \text{Global Type} & \xrightarrow{\text{projection}} & \text{Local Type} \\ \text{Code} & \xrightarrow{\text{type inference}} & \text{Local Type} \end{array}\right\} \quad \text{Compare the two}$$

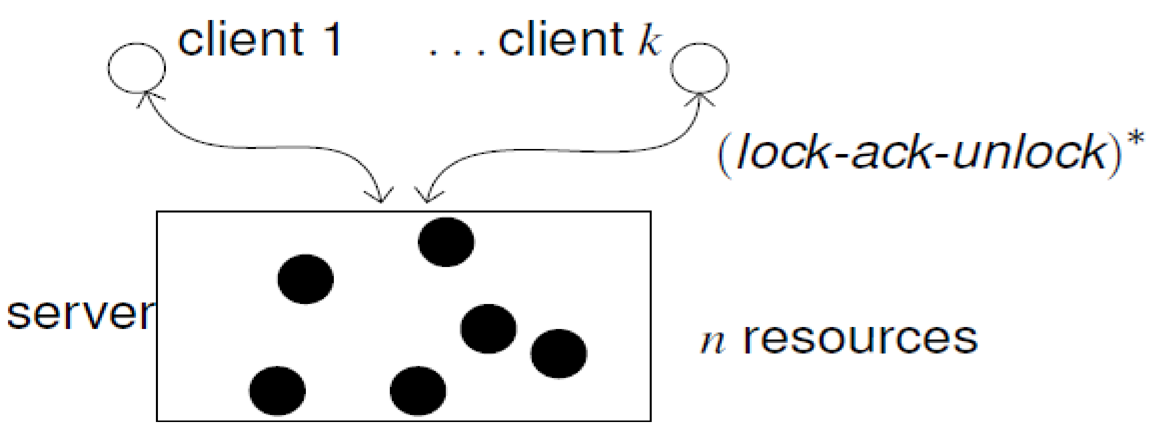
- *Reject messages not conforming to the type*
  - *Enforces safety and security properties.*

Example Session Type

Parameterization

Example (Limited resource locking)

$$\mathop{\Big\|}_{i=1}^{n} \left(\bigoplus_{j=1}^{k} \left(c_j \xrightarrow{lock_i} s \; ; \; s \xrightarrow{ack_i} c_j \; ; \; c_j \xrightarrow{unlock_i} s \right) \right)^{*}$$



Type Inference

- Observing previous patterns
  - In the example before, this is what happened three times in a row:

$$\mathsf{base} \xrightarrow{\mathit{move}} \mathsf{master} \; ; \; \mathop{\Big\|}_{i=1}^{n} \mathsf{master} \xrightarrow{\mathit{loc}} \mathit{UAV}_i \; ; \; [\mathsf{movement\ happens\ here}] \; ; \; \mathop{\Big\|}_{i=1}^{n} \mathit{UAV}_i \xrightarrow{\mathit{done}} \mathsf{master} \; ; \; \mathsf{master} \xrightarrow{\mathit{ok}} \mathsf{base}$$

  - **The fourth relocation sequence observed an abnormality between completion of the movement and sending the “done” message.**

Type Inference

- From code

```
Actor UAV
{
  msg loc(Point p)
  {
    master ! OK
    [ physical movement to p ]
  }
}
```

```
actor Master
{
  msg move
  {
    for(v in UAV)
      v ! loc(p)
  }
  msg done
  {
    if(++count == UAV_COUNT)
      base ! ok
  }
}
```

$$\text{base} \xrightarrow{move} \text{master} \ ; \ \overset{n}{\underset{i=1}{\|}} \text{master} \xrightarrow{loc} UAV_i \ ; \ [\text{movement happens here}] \ ; \ \overset{n}{\underset{i=1}{\|}} UAV_i \xrightarrow{done} \text{master} \ ; \ \text{master} \xrightarrow{ok} \text{base}$$

Questions and Answers

Thank you for all your attention

August 5th 2013

United States of America
Department of the Air Force
http://www.airforce.com

University of Illinois at Urbana-Champaign
http://www.illinois.edu

Information Trust Institute
http://www.iti.illinois.edu