

Feasibility Study and Early Experimental Results Towards Cluster Survivability



imm
immortalcomputing.com

Box Leangsuksun
Anand Tikotekar

Makan Pourzandi
Ericsson

Stephan Scott
ORNL

Ibrahim Haddad
OSDL



OAK RIDGE NATIONAL LABORATORY



Outline

- ❑ Motivation
- ❑ Current Scenarios
- ❑ An Approach: A proof-of-concept to Survivability - HA OSCAR + DSI
- ❑ Feasibility and Suitability of the Approach
- ❑ Future Work
- ❑ Summary
- ❑ References



Service Unavailability Impacts

- ❑ Losses of \$195K - \$58M with 3.5 hrs (Meta Group report, 2000)
- ❑ Enterprise/Shared Major computing resources- 7/24/365
- ❑ Critical HPC apps such as National Security (Home Land defense)
- ❑ Service provider Regulation/Mandate
 - ❖ FCC mandate (Class 5 local switch = 5 9's)
- ❑ April 2004 Security Breaches on TeraGrid
- ❑ Outages -> No Performance and No Functionality
- ❑ Losses time and opportunities
- ❑ Life-threatening

5/2/2005

3



Motivation

- ❑ Unplanned/Planned downtime in systems translates into huge loss due to security attacks/hardware failures/software failures
- ❑ Few approaches towards coherent and robust security in clusters/distributed systems
- ❑ Realizing 100% security is a myth and thus need for Effective recovery/Serviceability/Fault tolerance
- ❑ Above requirements are tantamount to an approach that leads to overall system Survivability, Dependability and Availability

5/2/2005

4



Current Scenario

- Security in clusters is largely based on securing individual nodes
 - Too weak for hackers and too strong for beginners
- Interoperability issues and maintainability issues.
- Cluster security is not tantamount to securing individual nodes, and differs from security on a single machine
- Existing Cluster management systems show minimal concern towards all important RASS
- Many HA systems such as Linux Failsafe, Kimberlite, HP Serviceguard, HA-OSCAR, lack a coherent security framework essential for system survivability

5/2/2005

6



What we mean by RASS

- Reliability(MTTF): *How fast it fails?*
- Availability: The probability that a system is up and kicking at any given point -> $MTTF / (MTTF + MTTR)$
- Serviceability: The ease with which Preventive and corrective maintenance can be performed on the system
- Security: State of being free from danger or injury

5/2/2005

7



RASS goal or Ineluctability of Failure

Reliability - Availability - Serviceability - Security

- ❑ A Chain is only as strong as its weakest link
- ❑ Miss one and you end up missing four
- ❑ No conflict of interest between parameters
- ❑ RASS encapsulates Survivability
- ❑ “RAS” parameters are provided through HA OSCAR (discussed next) and “S” is provided through DSI.

5/2/2005

8



HA-OSCAR (RAS) 1/3

- ❑ Open source clustering software toward Non-Stop services in HEC environment
- ❑ Combination of HA features and HPC capabilities
- ❑ First Field Grade HA Beowulf cluster combining High Availability and Critical self-healing services

5/2/2005

9



HA-OSCAR

2/3

- Component Redundancy : High Availability
- Self-Configuration : Cluster Management
- Adaptive Self Healing : Serviceability
- Service Level Fault Tolerance : Survivability
- Disaster Recovery : Serviceability

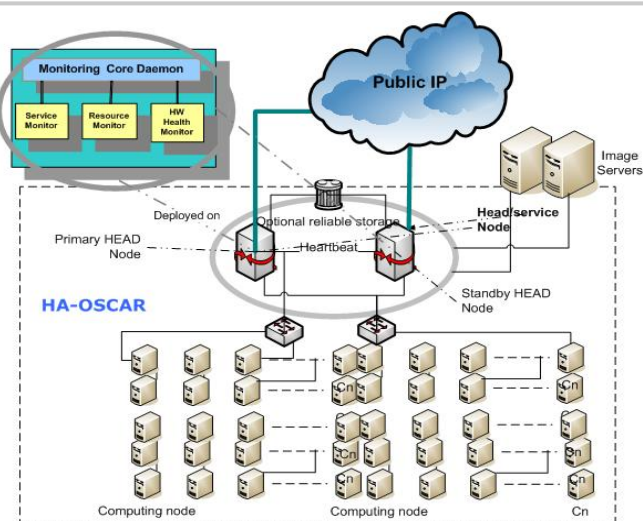
5/2/2005

10



HA-OSCAR

3/3



5/2/2005

11



Distributed Security Infrastructure (DSI)

- Developed for Cluster Environment
 - ✧ Fine grained, Flexible, Adaptable
- High level of abstraction for access control
 - ✧ Separating administrative, network, computation into different security zones
- **Process level + User level**
 - ✧ Kernel level module (DSM)
 - ✧ Real time checks based on the LSM hooks

5/2/2005

12



Distributed Security Infrastructure (DSI) Characteristics

- Coherent framework
 - ✧ Single server vs. cluster security are different animals
- Process level approach and more
 - ✧ Instead of all or nothing
- Pre-emptive security
- Transparent key management
- Dynamic security policy

5/2/2005

13



DSI – components (1/2)

- ▣ Security Server
 - ✦ Central point of management, policy holder
- ▣ Security Manager
 - ✦ Node based enforcement of policy
- ▣ Secure Communication Channel
 - ✦ Encrypted, authenticated. SS <-> SM
- ▣ Distributed Security Policy (DSP)
 - ✦ XML, rules spanning entire cluster

5/2/2005

14



DSI – components (2/2)

- ▣ Robust Security
 - ✦ Enforce rules at kernel level
- ▣ Distributed Security Module (DSM)
 - ✦ Set of kernel hooks implementing DSP via LSM
- ▣ Linux Security Module (LSM)
 - ✦ Enforces as part of DSI access control service
 - ✦ Integrated with SCC

5/2/2005

15



DSI Security Policy (DSP)

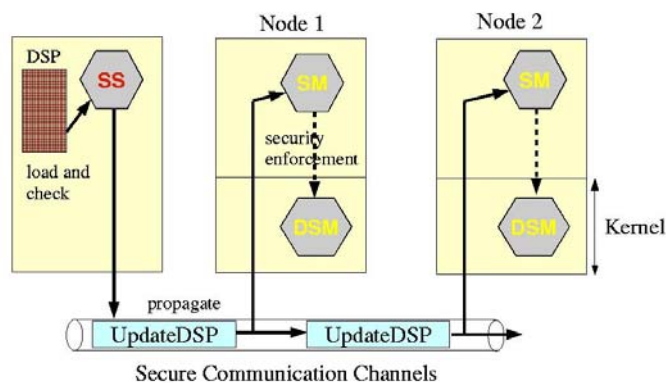
- ❑ Coherent security vision for entire cluster
- ❑ Maintained, Updated by Security Server
- ❑ Based on Domain Enforcement
- ❑ Expressed In XML
- ❑ Separation of Policy decision logic & Policy enforcement logic (Flask architecture)

5/2/2005

17



DSP Architecture



5/2/2005

18

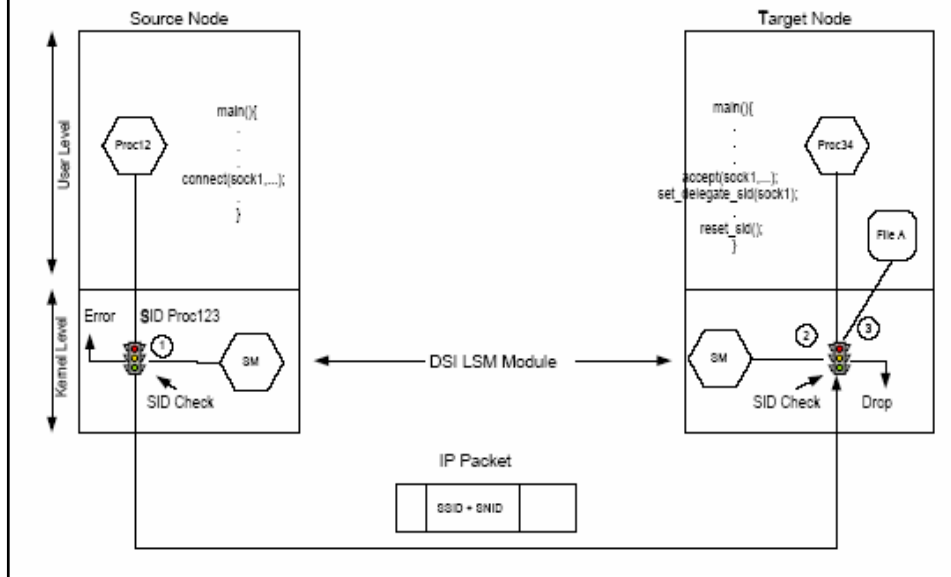


Sample DSP Rule

- ❑ `<class_PROCESS_Rule>`
- ❑ `<ScID>1</ScID>` // a process with ScID 1
- ❑ `<SnID>2</SnID>` // on node SnID 2
- ❑ `<allow>CREATE</allow>` // can create a
- ❑ `</class_PROCESS_Rule>` // process/call fork()

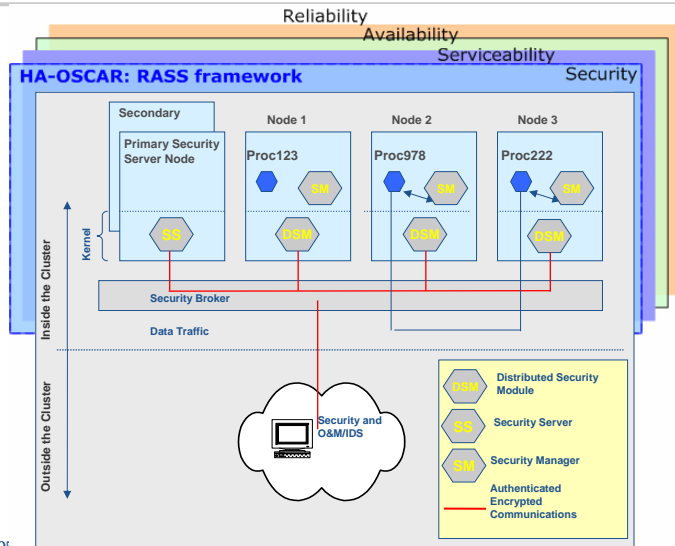
5/2/2005

19





Cluster Survivability Architecture



5/2/2005

23



Key Approach to our RASS framework

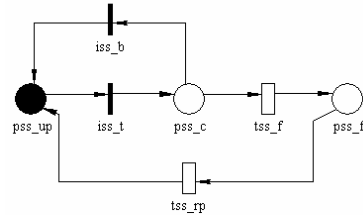
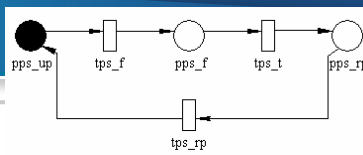
- Enforceable Kernel Security through DSM/DSP (DSI)
- Distributed Security Layer through DSI
- Serviceability layer to check important services including security services of DSI through HA-OSCAR
- Failover or Availability layer – HA-OSCAR monitoring and self-healing core
- Reliability/Dependability/Survivability proposed through the layered architecture

5/2/2005

24



Reality Checks



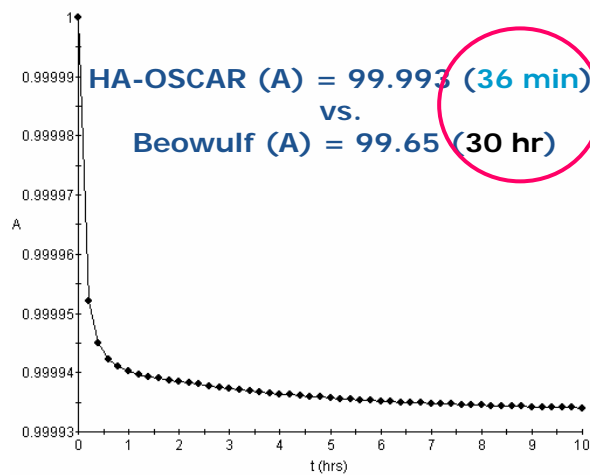
- But How much improvement?
 - The total uptime?
 - Performance?
- Analytical model and prediction
 - How many 9's? (downtime per/year)
 - Stochastic Reward Net using SPNP package from Duke U.

5/2/2005

25



Availability Improvement



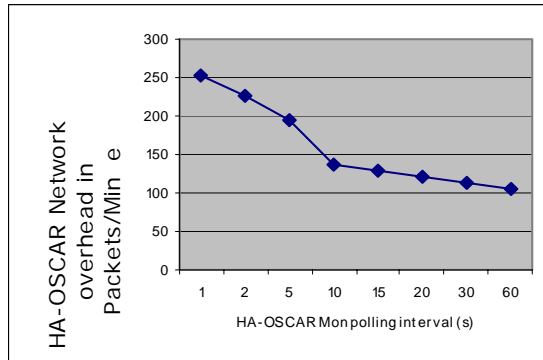
5/2/2005

26



Feasibility study - Overhead analysis

- DSI 0.3 and HA-OSCAR 1.0 beta on Linux cluster, Redhat 9.0, P4 2.4Ghz
- Overhead in Package According to polling Interval
- CPU usage between 0.9% to 1% by HA OSCAR Mon daemons



5/2/2005

27



More Benchmarking

Time in microseconds

Test type	Without DSI	With DSI	Overhead
Stat	1.92	1.94	1.0%
Open/Close	2.68	2.68	0%
Fork	92.81	93.58	0.82%
Exec	322.56	328.33	1.78%
Sh proc	2140.75	2150	0.43%
UDP	9.68	10.61	9.6%
RPC/UDP	17.66	18.7	5.9%
TCP	11.08	12.68	14.4%
RPC/TCP	23.42	24.3	3.75%

5/2/2005

28



Issues

- ❑ Few issues from OmniORB (httpds as future alternative)
- ❑ Distros conflicts (various gcc issues from OSCAR vs DSI 0.3) were overcome.
- ❑ DSM using LSM has an equivalent functionality of SELinux, NOT Good.

5/2/2005

29



Future Work

- ❑ DSP Enhancement/Ease of Expression
- ❑ Porting to SELinux Functionality instead of DSM
- ❑ Localization of Damages
- ❑ Intelligent Recovery Policy using event analysis
- ❑ More Application opportunities for our RASS framework

5/2/2005

30



Summary

- ❑ Existing security solutions adhere to “Individually Repealing attacks”
- ❑ Demonstrated RASS based Survivability
- ❑ Minimal performance overhead, thus keeping alive HPC proposition
- ❑ Cluster RASS can be achieved with acceptable overheads
- ❑ Satisfying goals of Mission Critical Systems

5/2/2005

31



Related Links

- [1] HA-OSCAR: <http://xcr.cenit.latech.edu/ha-oscar/>
- [2] DSI : <http://disec.sourceforge.net>
- [3] Open Systems Lab: <http://www.linux.ericsson.ca>

5/2/2005

32



How do we plan on doing it ?

- ❑ SELinux
- ❑ Accepted into mainstream kernel
- ❑ Default on the 2.6.xx series
- ❑ However, SELinux is for end systems

- ❑ **GOAL:** Combine the granularity of SELinux and distributive characteristic DSI into a cluster security solution

5/2/2005

33



Conversion Table

DSI	SELinux
each XML node (each security class)	domain type (everything is a type) - specific allow cmds
PROCESS	<process where> <process target> : process { perms }
Binary	type \$_exec_t file_type, sysadmfile, exec_type;
TRANSITION	type_transition \$_t \$_exec_t :process \$_t; <what> <through what> <to what> allow 1_t 4_t :process transition; <from> <to> <how>
Nodes	type node \$_t, node_type; <what> <type>
NETWORK	<from> <to> :rawip_socket <which node> netif_type :netif { perms };
SOCKET	tcp_socket,udp_socket,unix_stream_socket <from> <to> :<type of socket> { perms } <from> <to node> :node { perms }
Allow/deny	allow/neverallow <from> <to> <type> { perms }

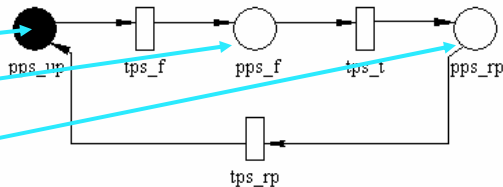
5/2/2005

34

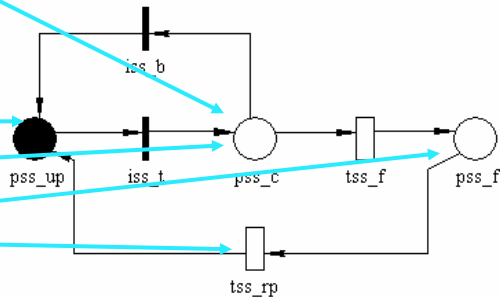


SPN Model

- P Server up
- P Server down
- Failover
- P server repair
- Failback



- S is up and ready
- S takes control
- S Server down
- S repair



5/2/2005

35