47th Meeting of IFIP WG10.4
Puerto Rico – Jan 2005

Non-intrusive Middleware for Continuity of Service: Protection Against System Failures

Marc Rougier - Meiosys

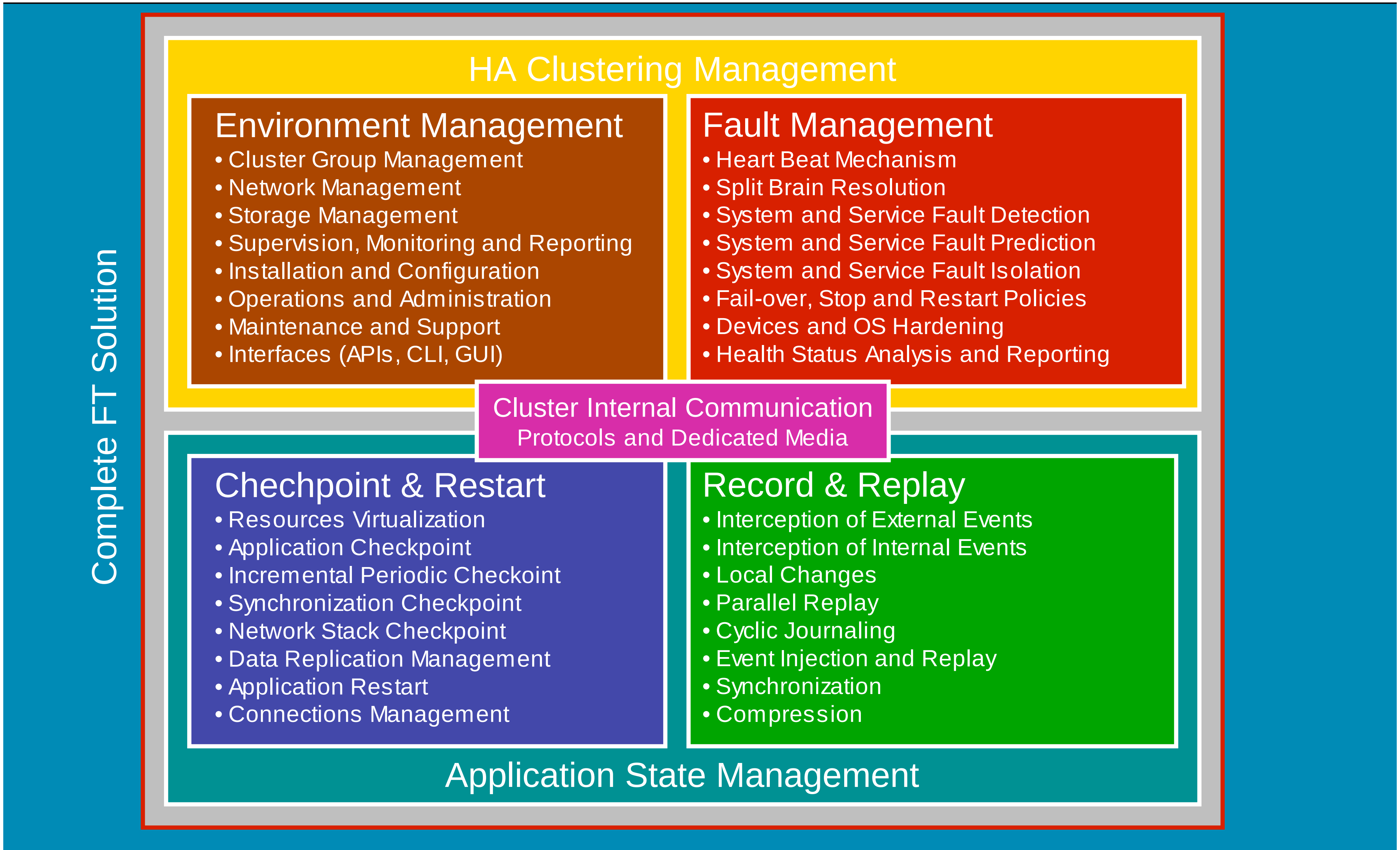
About Meiosys

- Independent Software Vendor, founded in 2000
- 35 people, 25 engineers in Toulouse, France and Palo Alto, CA, USA
- Genes are in middleware for distributed, life-critical systems
- Develops linux and Unix-based middleware to increase flexibility and dependability of commodity platforms
- Main topic of R&D today is Record and Replay technology for Fault Tolerance

Meiosys FT R&D Objectives

- Mission is to increase the service uptime (at an acceptable cost)
- Focus is to protect against system failures
  - Solution provides a dependable infrastructure…
  - But does not solve all problems (software bugs, human errors, etc)
- Approach is based on
  - Hardware redundancy and
  - Dedicated middleware maintaining operational and back-up systems in-sync
  - Active-Passive and Active-Active mode
- Main challenges
  - Application-transparent: no modification, re-compile nor re-link of the application
  - Runs on commodity equipment (off-the-shelf servers)
  - Performances impact needs to be “acceptable”
  - Needs to be applicable to commercial ISVs applications (DBMS, AS, ERP, etc), new applications (J2EE) and legacy applications
- Main problem: *the non deterministic nature of linux / Unix*

FT Solution: High Level Components



Complete FT Solution

HA Clustering Management

Environment Management

- Cluster Group Management
- Network Management
- Storage Management
- Supervision, Monitoring and Reporting
- Installation and Configuration
- Operations and Administration
- Maintenance and Support
- Interfaces (APIs, CLI, GUI)

Fault Management

- Heart Beat Mechanism
- Split Brain Resolution
- System and Service Fault Detection
- System and Service Fault Prediction
- System and Service Fault Isolation
- Fail-over, Stop and Restart Policies
- Devices and OS Hardening
- Health Status Analysis and Reporting

Cluster Internal Communication

Protocols and Dedicated Media

Chechpoint & Restart

- Resources Virtualization
- Application Checkpoint
- Incremental Periodic Checkoint
- Synchronization Checkpoint
- Network Stack Checkpoint
- Data Replication Management
- Application Restart
- Connections Management

Record & Replay

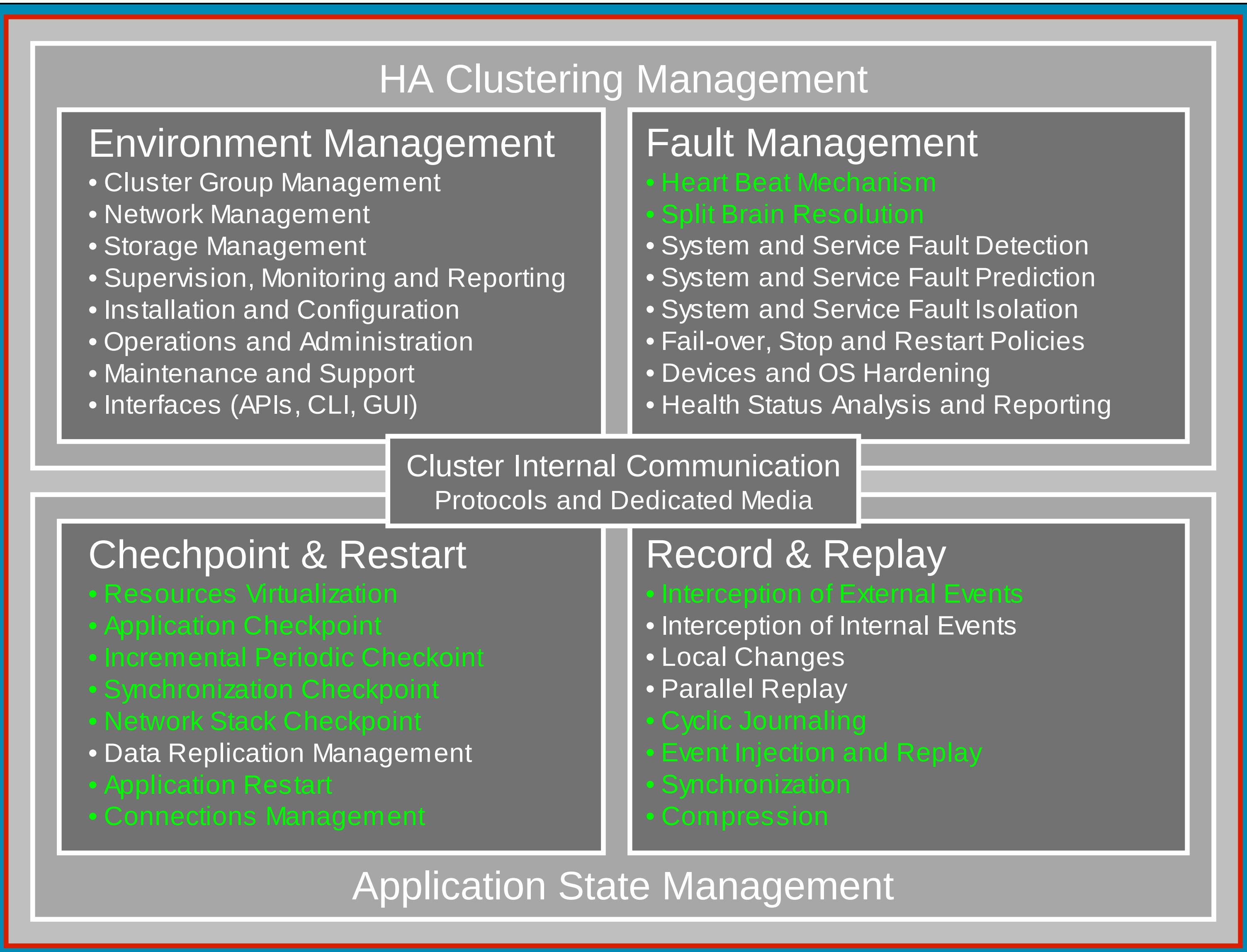
- Interception of External Events
- Interception of Internal Events
- Local Changes
- Parallel Replay
- Cyclic Journaling
- Event Injection and Replay
- Synchronization
- Compression

Application State Management

Meiosys FT R&D: Current Status

MEIOSYS
Optimize and Protect

Complete FT Solution



HA Clustering Management

Environment Management

- Cluster Group Management
- Network Management
- Storage Management
- Supervision, Monitoring and Reporting
- Installation and Configuration
- Operations and Administration
- Maintenance and Support
- Interfaces (APIs, CLI, GUI)

Fault Management

- Heart Beat Mechanism
- Split Brain Resolution
- System and Service Fault Detection
- System and Service Fault Prediction
- System and Service Fault Isolation
- Fail-over, Stop and Restart Policies
- Devices and OS Hardening
- Health Status Analysis and Reporting

Cluster Internal Communication

Protocols and Dedicated Media

Chechpoint & Restart

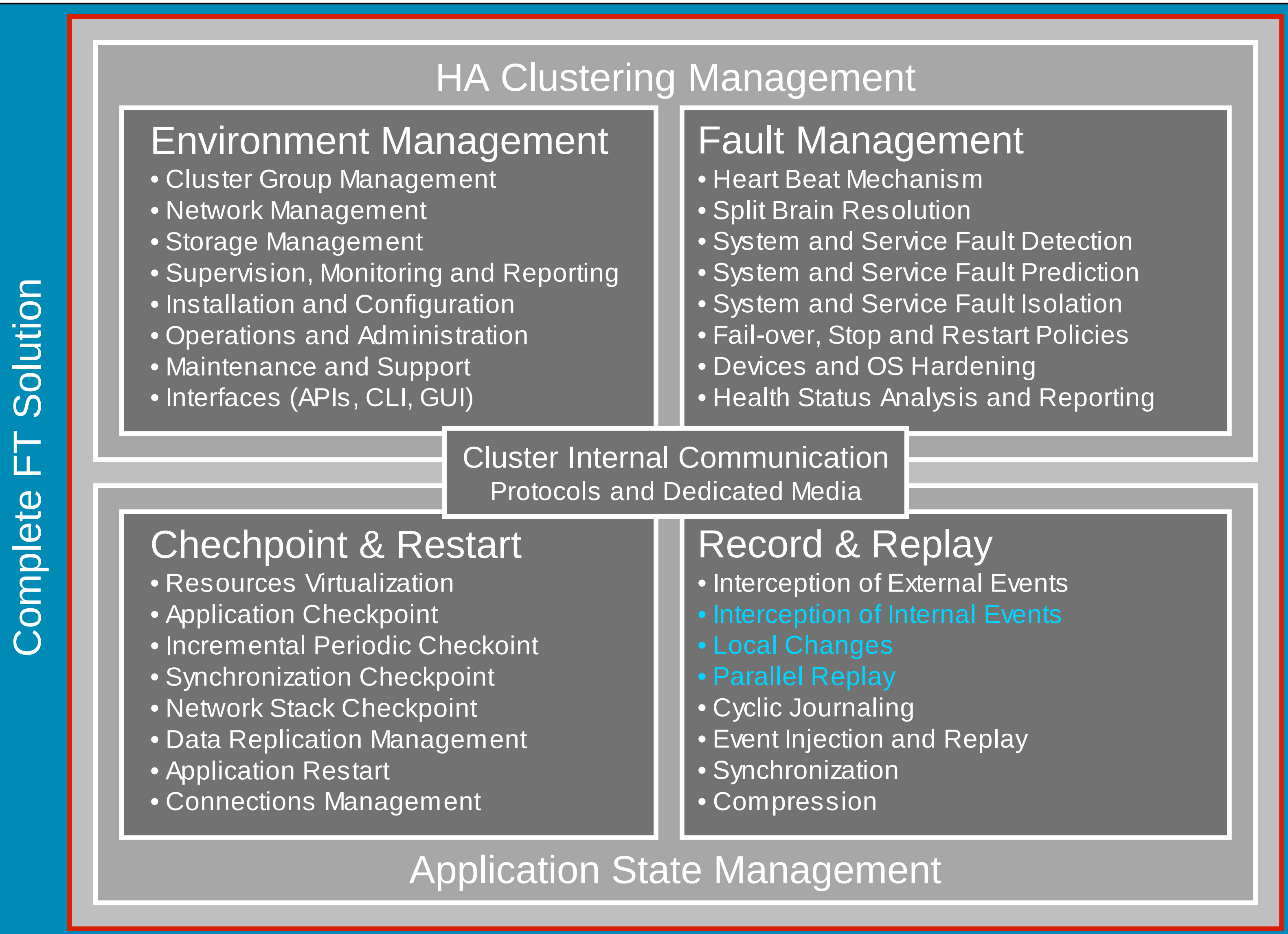
- Resources Virtualization
- Application Checkpoint
- Incremental Periodic Checkoint
- Synchronization Checkpoint
- Network Stack Checkpoint
- Data Replication Management
- Application Restart
- Connections Management

Record & Replay

- Interception of External Events
- Interception of Internal Events
- Local Changes
- Parallel Replay
- Cyclic Journaling
- Event Injection and Replay
- Synchronization
- Compression

Application State Management

Meiosys FT R&D: Current Focus



Complete FT Solution

HA Clustering Management

Environment Management

- Cluster Group Management
- Network Management
- Storage Management
- Supervision, Monitoring and Reporting
- Installation and Configuration
- Operations and Administration
- Maintenance and Support
- Interfaces (APIs, CLI, GUI)

Fault Management

- Heart Beat Mechanism
- Split Brain Resolution
- System and Service Fault Detection
- System and Service Fault Prediction
- System and Service Fault Isolation
- Fail-over, Stop and Restart Policies
- Devices and OS Hardening
- Health Status Analysis and Reporting

Cluster Internal Communication

Protocols and Dedicated Media

Chechpoint & Restart

- Resources Virtualization
- Application Checkpoint
- Incremental Periodic Checkoint
- Synchronization Checkpoint
- Network Stack Checkpoint
- Data Replication Management
- Application Restart
- Connections Management

Record & Replay

- Interception of External Events
- Interception of Internal Events
- Local Changes
- Parallel Replay
- Cyclic Journaling
- Event Injection and Replay
- Synchronization
- Compression

Application State Management

Technology Modules

Interfaces

Enables integration with third-party components in the data center
(e.g., Tivoli, OpenView, Unicenter, HA clustering solution, billing systems, etc)

Policy Engine

Takes actions according to reported data; actions can be driven by optimization of resources (TCO), performances or uptime (SLA)

Monitoring

Captures and reports status data related to the behavior and health of the system and of the applications

Relocation

Orchestrates the mobility of the state files (mediation with the management layer, check of nodes consistency, N-stage migration with hand-checks)

R&R

Records and Replay all events which modify the application state
(external messages and internal non deterministic events)

Checkpoint

Captures in a « state file » all states constituting the run-time of the applications
(memory, IPCs, kernel states including TCP stack, etc)

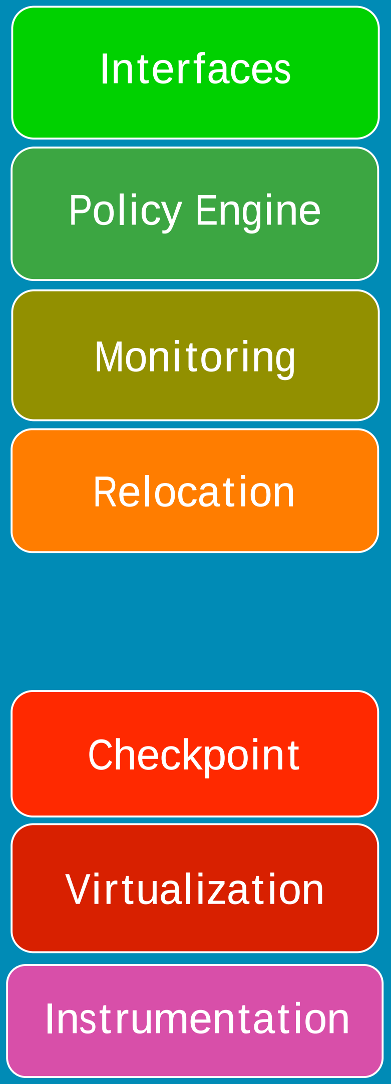
Virtualization

Maintains a near real-time view of the application structure in a « container » and subsitutes local IDs by relocatable IDs (e.g. PID, Sys V IPC IDs, etc)

Instrumentation

Techniques to interact with applications at run time
(e.g. interposition agents, kernel APIs, syscall injection)

Technology Modules



Interfaces

Policy Engine

Monitoring

Relocation

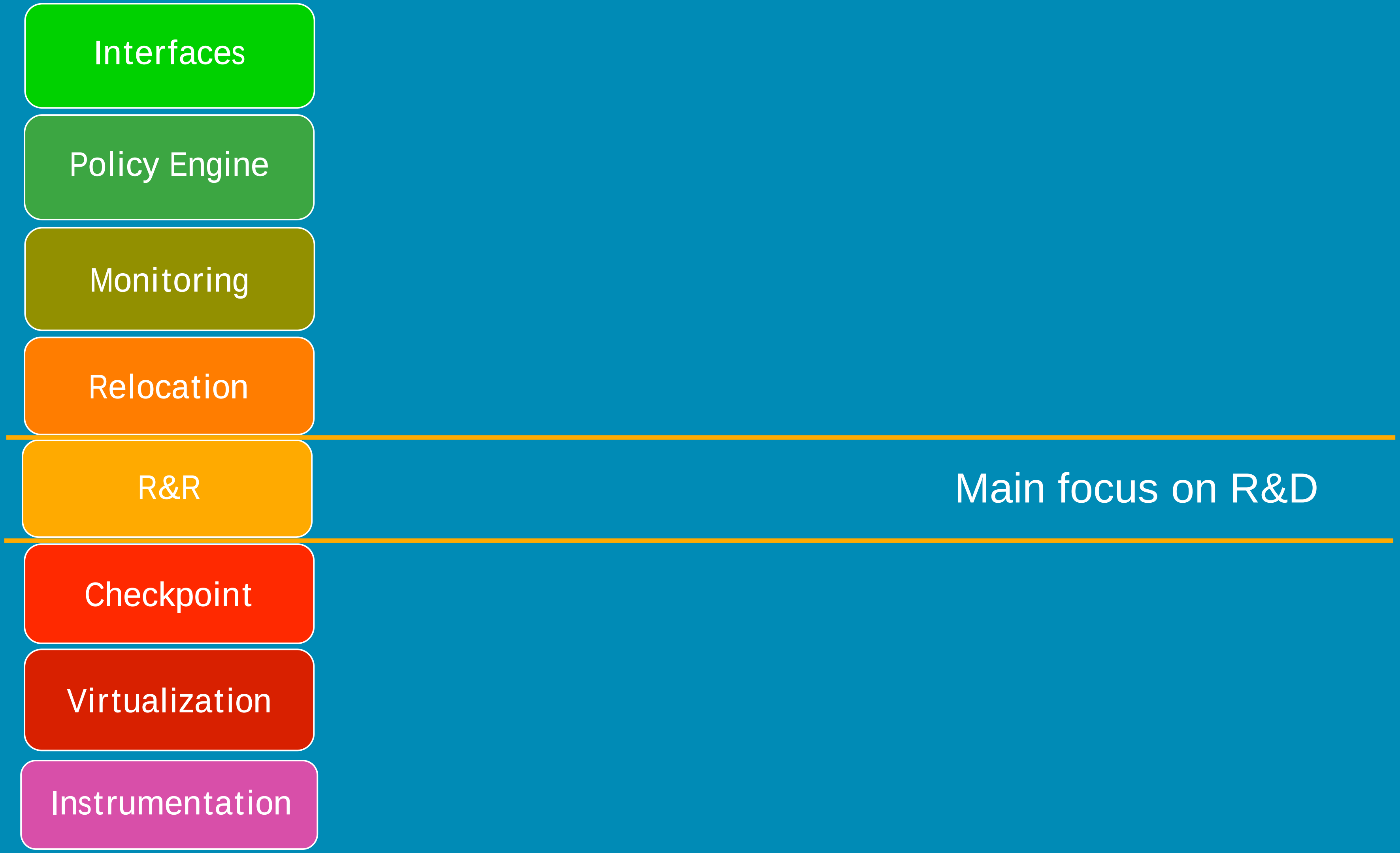
Checkpoint

Virtualization

Instrumentation

- Completed (and shipping)
- Enables dynamic, on-demand workload placement
- Maintains full states and network connections
- Thin virtualization layer (<1% runtime overhead)
- Granularity = application-level
- Stateful Application Relocation can be triggered by:
    - Resource optimization policies (consolidation)
    - Performance optimization policies (scale up)
    - High Availability policies (predictive fail-over)

Technology Modules



Interfaces

Policy Engine

Monitoring

Relocation

R&R

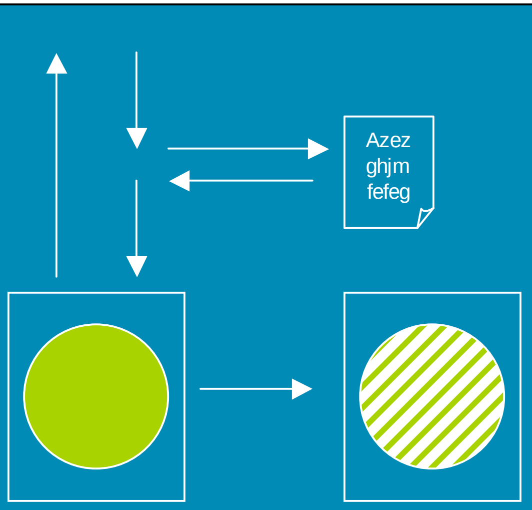
Main focus on R&D

Checkpoint

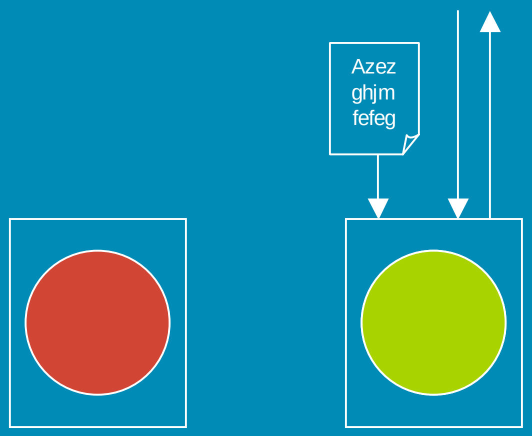
Virtualization

Instrumentation

Active-Passive Mode: Enables N+K

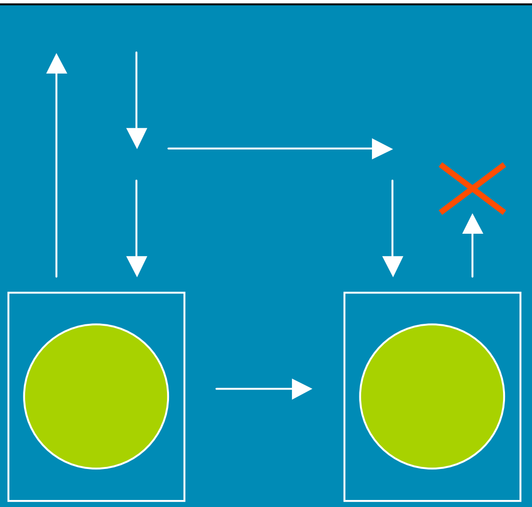


- A passive clone is maintained nearly up-to-date on the back-up node through incremental periodic checkpoint (in sync with the journal)
- Events are logged synchronously on the back-up node in a revolving journal
- Clone is stateful and includes TCP connections

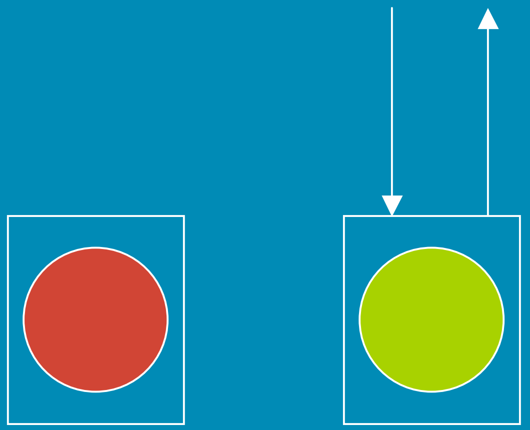


- Upon detection of outage of main node, journal is replayed on backup, so as to bring the clone up-to-date, in sync with external world (no messages are output during replay)
- Then communications are re-established with external world via migrated connections
- During replay, incoming messages are « on hold » (TCP flow control property)

Active-Active Mode: Faster Switch-Over



- Initial synchronization is achieved through a checkpoint
- Events are forwarded optimistically to the back-up node, on the fly
- Events are processed on both nodes but only the master sends messages to external world



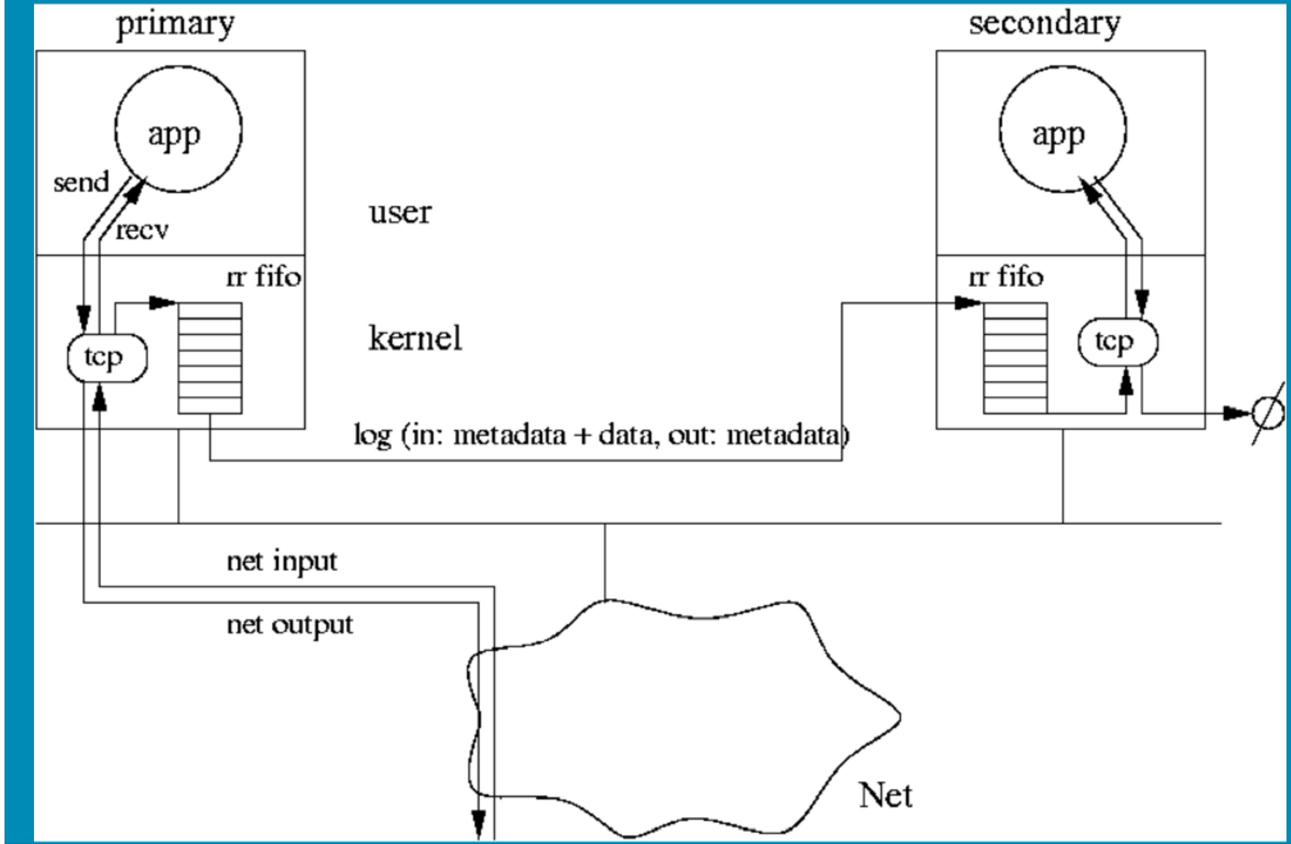
- Upon detection of outage of main node, current log is flushed and IOs are switched from shadow mode to operational mode
- Backup node immediatly resumes operations (sub-second switch-over)
- Order to switch-over can come from an external system (not necessarily a fault)

The Challenge of R&R: Non Determinism

- A State can be modified by external and internal events
- External Non Determinist Events (ENDE):
  - Inputs from network (TCP), or shared storage
  - Medium frequency (up to 10 Khz), medium volume (1-10 KB / event)
- Internal Non Determinist Events (INDE):
  - Non-determinist conditions due to OS or HW concurrency:
    - SHM access ordering , FS access order, IPCs, signals, I/Os
  - Random conditions:
    - Date (timestamps), timers, random numbers
  - High frequency (up to 10 Mhz), low volume (~ 10 B / event)
  - Internal NDEs between last external NDE and crash time can be lost
- The challenge is to Record and Replay these events deterministically, to maintain service integrity

R&R of External Events: TCP

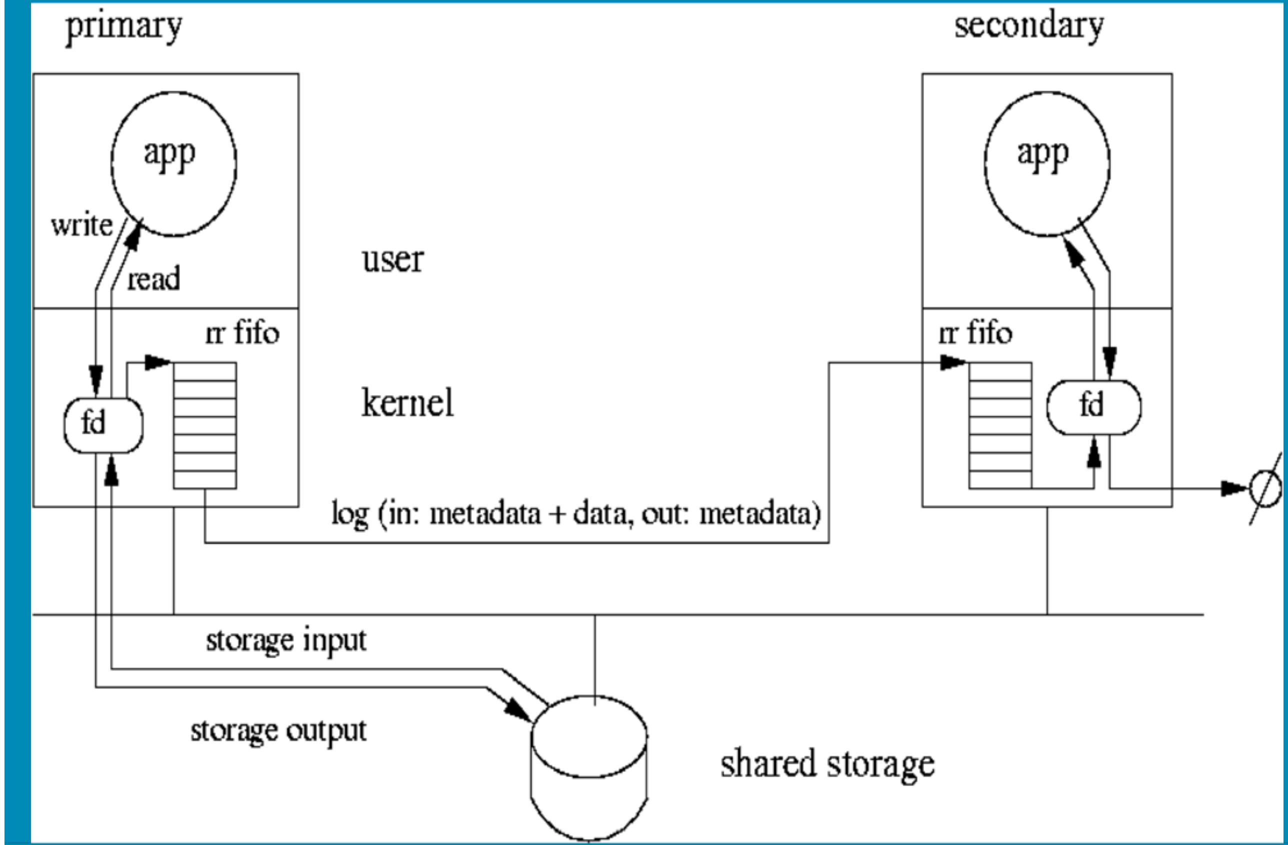
- Both nodes have the same virtual IP address. Only primary is visible.
- On primary: network input data, and connection metadata are logged on the fly to secondary.
- On secondary: network output disabled. Shadow sockets are feed and maintained up-to-date from the log and active application replica.



- Switch-over: at end of log, secondary takes over network physical access. Shadow sockets are ready to take over.
- Stand-by reinsertion: TCP sockets are checkpointed and cloned as part of process resources.
- No loss of in-flight messages: ACK’ed by primary only after logging. If crash during logging, retransmit by TCP.

R&R of External Events: Shared Storage

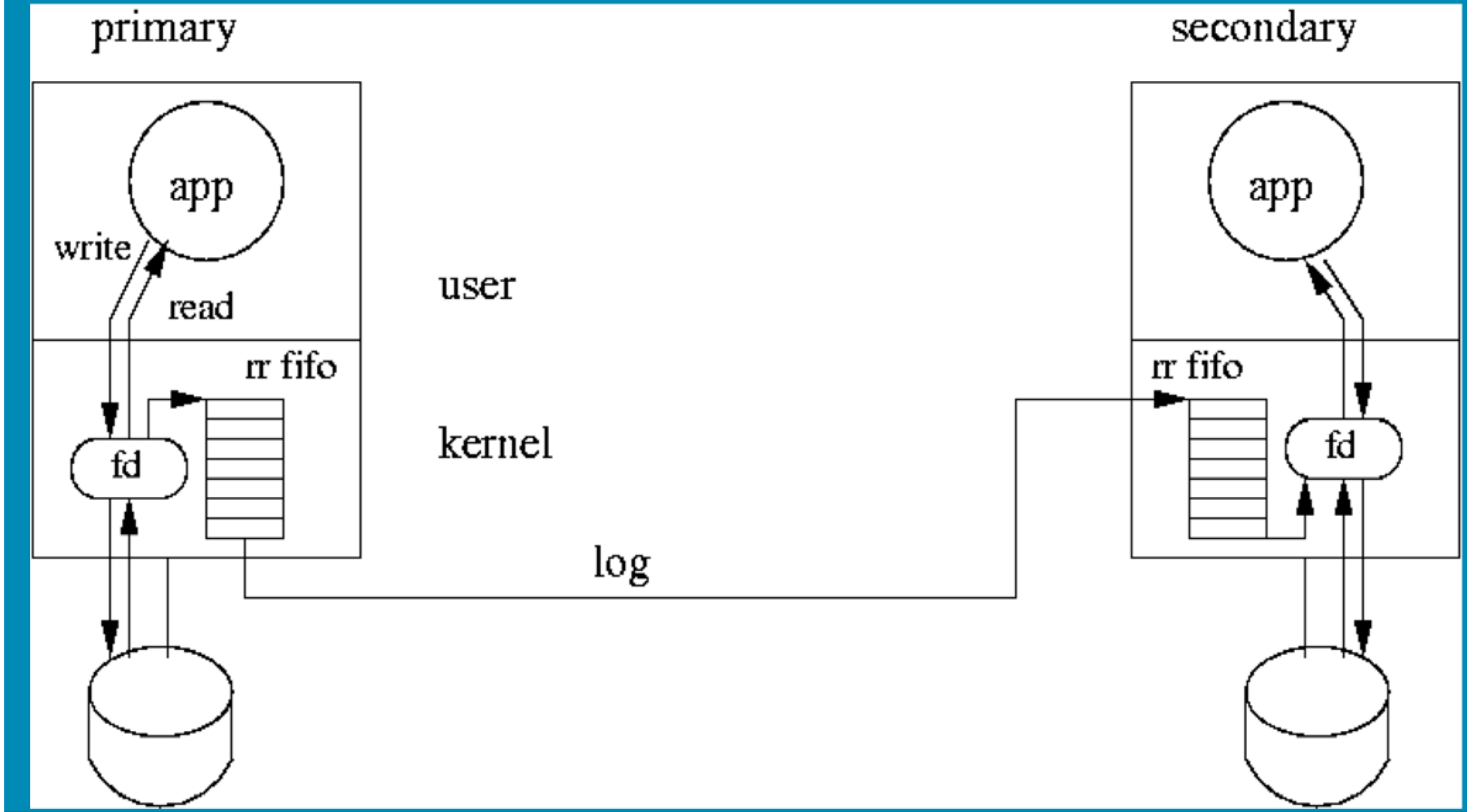
- Only the primary node has physical access to the shared storage
- On primary: inputs and system calls metadata are logged to secondary on the fly
- On secondary: output to storage is disabled
- Storage metadata (shadow file descriptors) are updated on the fly by active application replica and log



- At switch-over: secondary enables access to storage (procedure depends on type of storage)
- Shadow file descriptors mapped on real storage
- Reinsertion of standby: nothing to be done

R&R of External Events: Unshared Storage

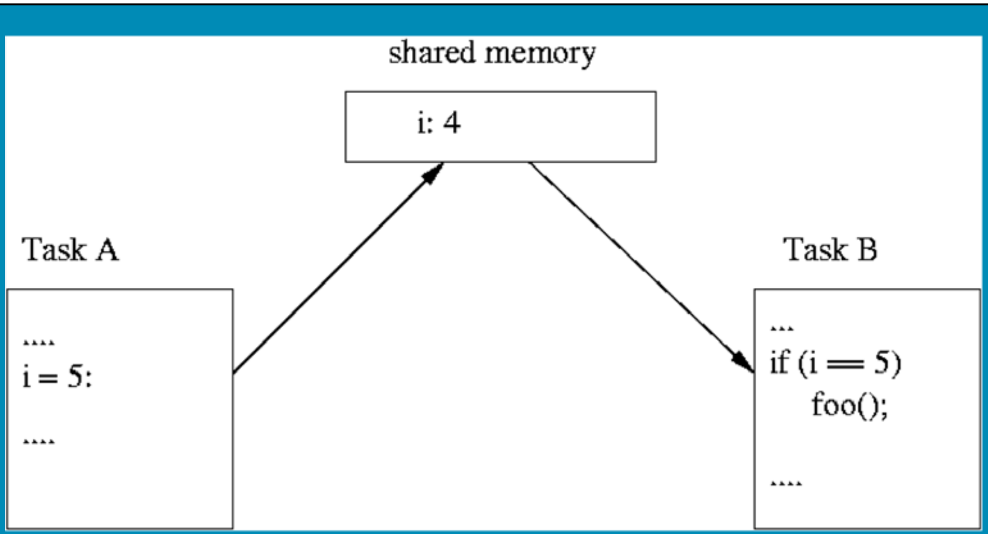
- Storage considered as a local resource
- Only storage access system calls metadata are logged
- At switch-over: the storage is already operational
- Reinsertion of stand-by: requires filesystem snapshot and replication capabilities



R&R of Internal Events: “Easy” Cases

- I/O-related System calls (non deterministic size)
  - Record and Replay the behavior (number of bytes)
  - Or change behavior locally (“semantic change”) if more efficient (force number of bytes, hence reducing amount of data to be logged)
- I/O Multiplexing (non deterministic ordering)
  - Record and Replay the behavior (ordering)
  - Or change behavior locally (“semantic change”) if more efficient (force ordering, hence reducing amount of data to be logged)
- Date, Timestamps, Random numbers
  - Must be Recorded and Replayed

R&R of Internal Events: “Difficult” Cases



- SHM access:
  - Task A and B running on 2 parallel CPUs in SMP
  - Execution result depends on the ordering of SHM access by A and B
  - Race condition is arbitrated at physical level (CPU-MEM bus controller), beyond the reach of kernel
  - If ordering could be detected, logging each access would multiply unitary cost by 1000 (ref. works by Bacon & Goldstein – Berkeley and IBM Watson, on snooping the CPU-memory bus with specific hardware technology)
- Signal delivery
  - Task A sends a signal to task B
  - Crash occurs after task B receives the signal on Operational node but before task B receives the signal on back-up node
  - Task B needs to receive signal at the same instruction on back-up node

R&R of Internal Events: “Difficult” Cases. Approach: Repeatable Scheduling

- Repeatable Scheduling
  - Definition: ability to reproduce task interleaving at instruction level
  - If a task receives the same interrupts at the same execution points, it will reproduce the same outputs
  - Addresses R&R of several INDEs: signals, SHM, IPCs
  - Transparent to applications (kernel-level solution)
  - BUT:
    1. **It assumes that instruction counters are reliable… which is (generally) false**
    2. **It is not applicable to SMP: does not address hardware parallelism**

- Repeatable Scheduling on SMP architectures with reliable counters
  - Modify resource access control to implement exclusive access during scheduling slice
  - Each CPU logs its scheduling activity
  - Shared resource access log used for global ordering
  - Requires two new algorithms:
    - **Reliable Instructions Counter**
    - **Exclusive SHM Access**

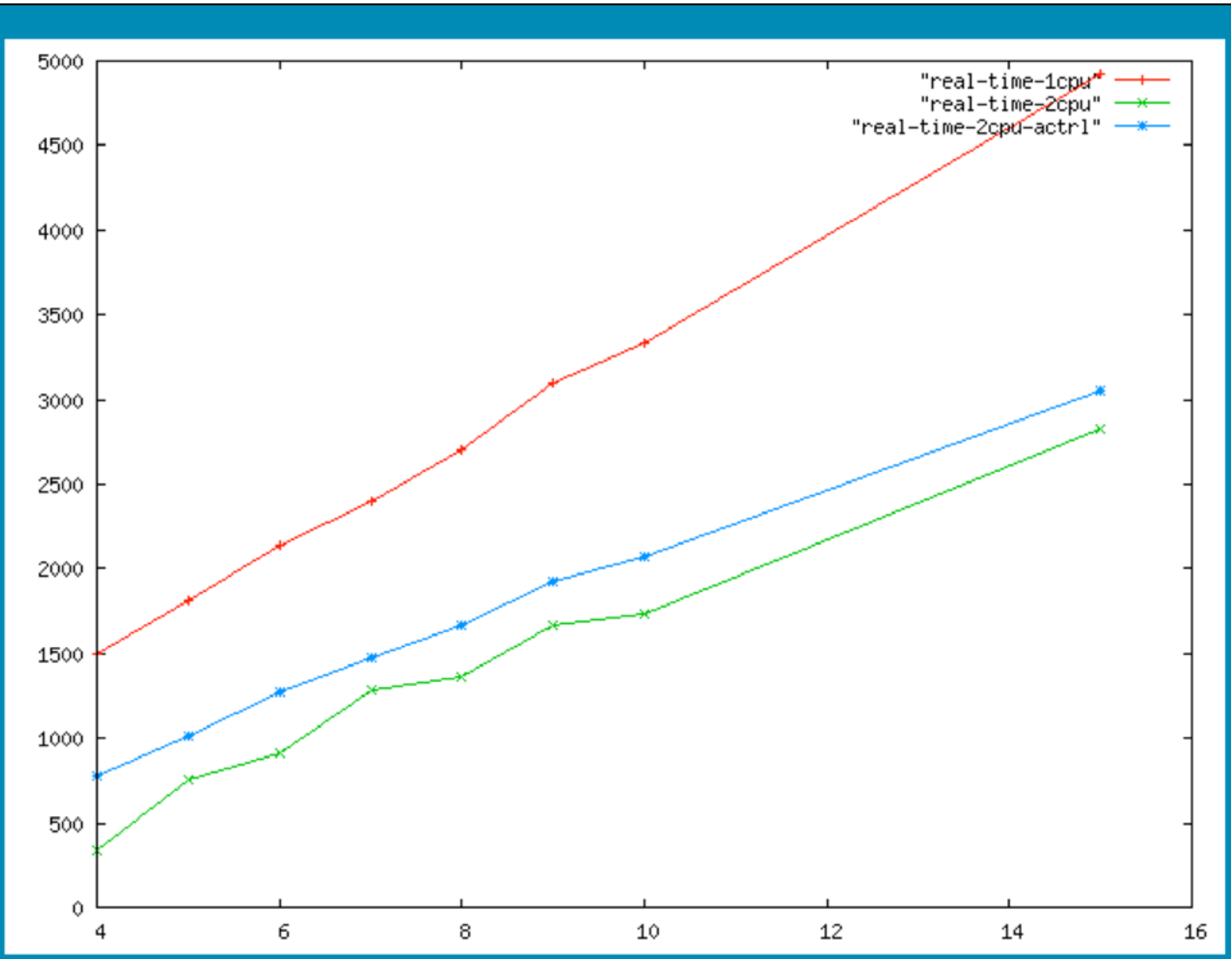
Reliable Instruction Counters

- Implement reliable instruction counting mechanism to complement repeatable scheduling on SMP architectures:
  - Hardware counters are available on modern CPUs, with negligible overhead
  - BUT not accurate: count of instructions impacted by pipelining, HW interrupts and exceptions, latency of overflow interrupts, micro-architecture optimizations
  - Forcing the CPU to produce precise instructions count makes it 25 times slower
  - Our approach: an additional software layer brings accuracy at instruction granularity level, compensating hardware inaccuracy
    - Software layer uses breakpoints to stop tasks at the exact location at Replay. Implements a reliable light weight CPU state checksum to handle closed loops
    - Scheduler's routines managing context switch have been extended
    - Record includes capture of signal delivery position
    - Enables to record on N CPUs and replay on M, whatever N and M ("logical CPU")

Exclusive SHM Access Control

- Implement exclusion mechanism to complement repeatable scheduling on SMP architectures:
  - Provides elected task with exclusive access to each shared memory page, for its scheduling period
  - Access control implemented by extending memory protection and paging mechanisms of MMU at kernel level
  - Allows to block a task if it accesses “in-use” SHM, freeing the slot for other work
  - Remove race conditions at user level
  - Allows reproducible SHM access at very low performance cost in SMP

Exclusive SHM Access Control and Reliable Instruction Counters: Performance Overhead



Performance hit less than 10%, scales gracefully with number of processes

Current Status and Next Steps

- Current Status:
  - On-demand stateful application relocation :
    - Works with transactional apps (Oracle, Weblogic) under heavy load
    - Contributes to increasing uptime thanks to predictive stateful fail-over triggered by fault management systems (system-level and application-level)
  - Active-Passive and Active-Active frameworks, with R&R of TCP and basic logging and fault detection mechanisms; sub-second switch-over
  - Reliable Instructions Counter algorithm
  - Exclusive SHM Access Control algorithm
- Next Steps:
  - Integration of all NDEs into Active-Active framework
  - Integration of a high performance logging infrastructure
    - Low latency interconnect and dedicated protocol
    - Optimization (cached logging “TCP-out committed ”, null logging, etc)
  - Full scale performance benchmarks

Thanks you for your attention

www.meiosys.com