

# XtreemOS



*Enabling Linux  
for the Grid*

## XtreemOS Grid Checkpointing

John Mehnert-Spahn

Heinrich-Heine University Duesseldorf, Germany

12<sup>th</sup> September 2009, Oxford, UK



Information Society  
Technologies

*XtreemOS IP project*

*is funded by the European Commission under contract IST-FP6-033576*





- Introduction
- Fault tolerance - node-level
- XtreemOS Grid Checkpointing architecture
- Discussion

joint work by:

**John Mehnert-Spahn, Eugen Feller, Michael Schoettner  
(UDUS, Duesseldorf, Germany) and  
Thomas Ropars and Christine Morin  
(Inria, Rennes, France)**





## New York



## London



## Beijing



## Sydney





## Virtual Organisations

New York



London



Beijing



Sydney



VO1

VO2

VO3

VO4






### New York



 Job unit A1


### London



 Job unit A2


### Beijing



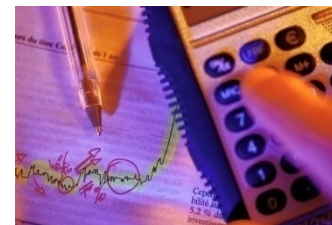
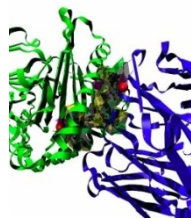
 Job unit A3

### Sydney



 Job unit A4

VO4 : user John: Job A





## Failures

New York



Job unit A1

London



Job unit A2

Beijing



Job unit A3

Sydney



Job unit A4

RESET





## Fault Tolerance

- **Virtual nodes**
- **Distributed servers**
- **P2P infrastructure**

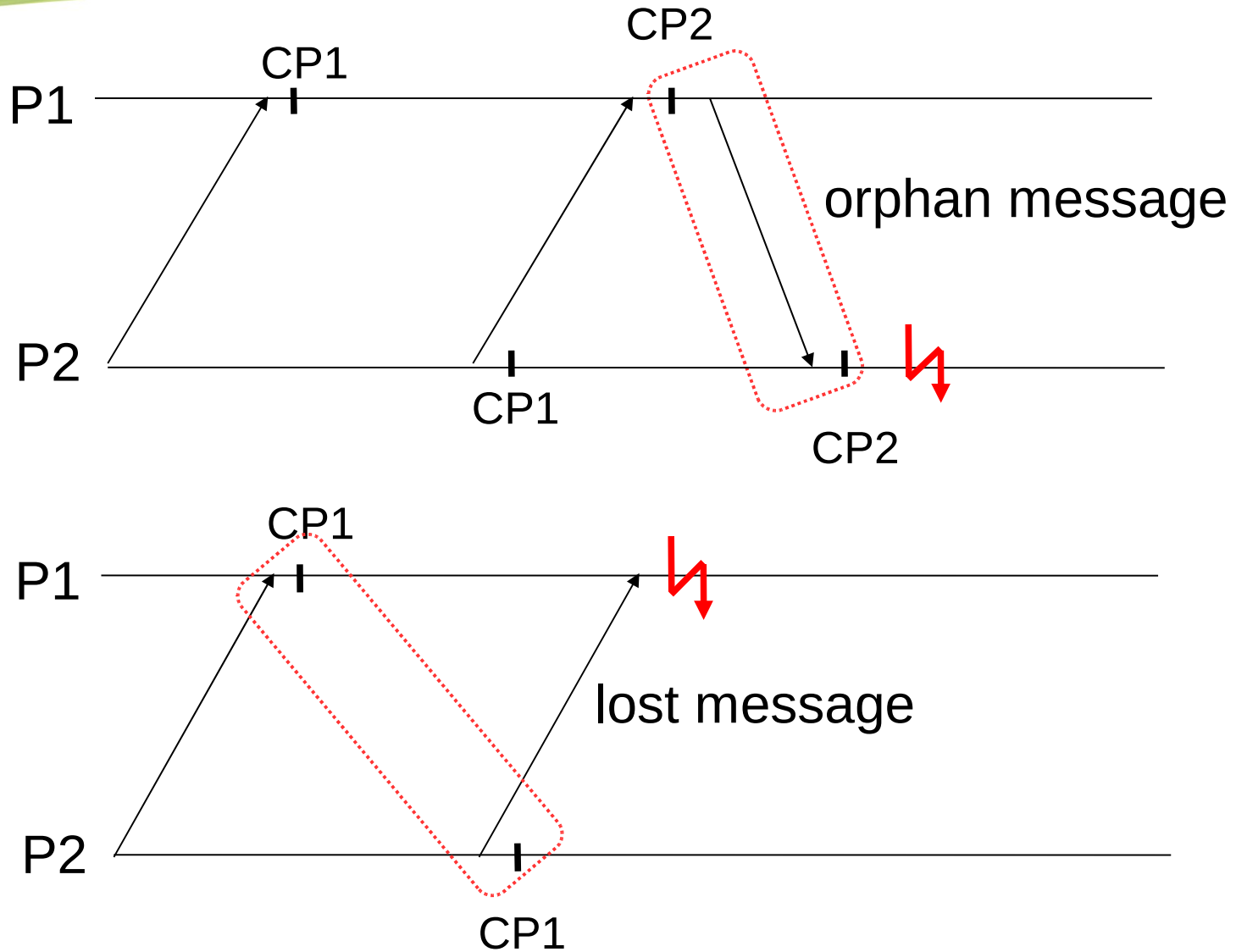




- **Main strategies:**
  - **backward error recovery**
  - **forward error recovery**
  
- **Backword Error Recovery – Cp. Protocols**
  - **checkpointing-based**
  - **message logging-based**
  - **combinations**
  
- **Checkpointing Strategies**
  - **full, incremental, concurrent**



# FT - lost vs. orphan messages





### ■ Coordinated Checkpointing

- synchronisation
- disk space, cp file removal
- consistency ensured at checkpoint
- for migration

### ■ Uncoordinated Checkpointing

- recovery line calculation, rollback, domino effect
- disk space, cp file removal
- consistency ensured at recovery





- **Application**
  - application implements checkpointing itself
- **Library**
  - application linked against checkpointer library
  - application transparent
- **Kernel**
  - OS supports checkpointing/restart
  - application transparent
- **Virtual machines**
  - **Transcendent Memory and Linux, D. Magenheimer et al., Oracle Corp, Linux Symposium, Canada, 2009**





# FT: Existing Checkpointers

Condor DMTCP & MTCP  
 Epckpt BLCR  
 UCLiK KMU  
 CoCheck TICK  
 VMADump MCR  
 LAM/MPI&BLCR CHPOX  
 Ckpt DCR  
 LinuxSSI zap  
 OpenVZ CRAK  
 SCore CLIP  
 Linux-native libckpt  
 Dynamite tmPVM  
 VMWare player





- **Abstraction level**
- **Checkpointing (in)capabilities**
  - **System vs. process states**
    - **Devices, GPU, ...**
  - **Distributed checkpointer?**
- **Checkpoint protocol supported**
- **Isolation and virtualization type**
  - **Framework or stand-alone**
    - **MPI, PVM**
  - **Callback support**



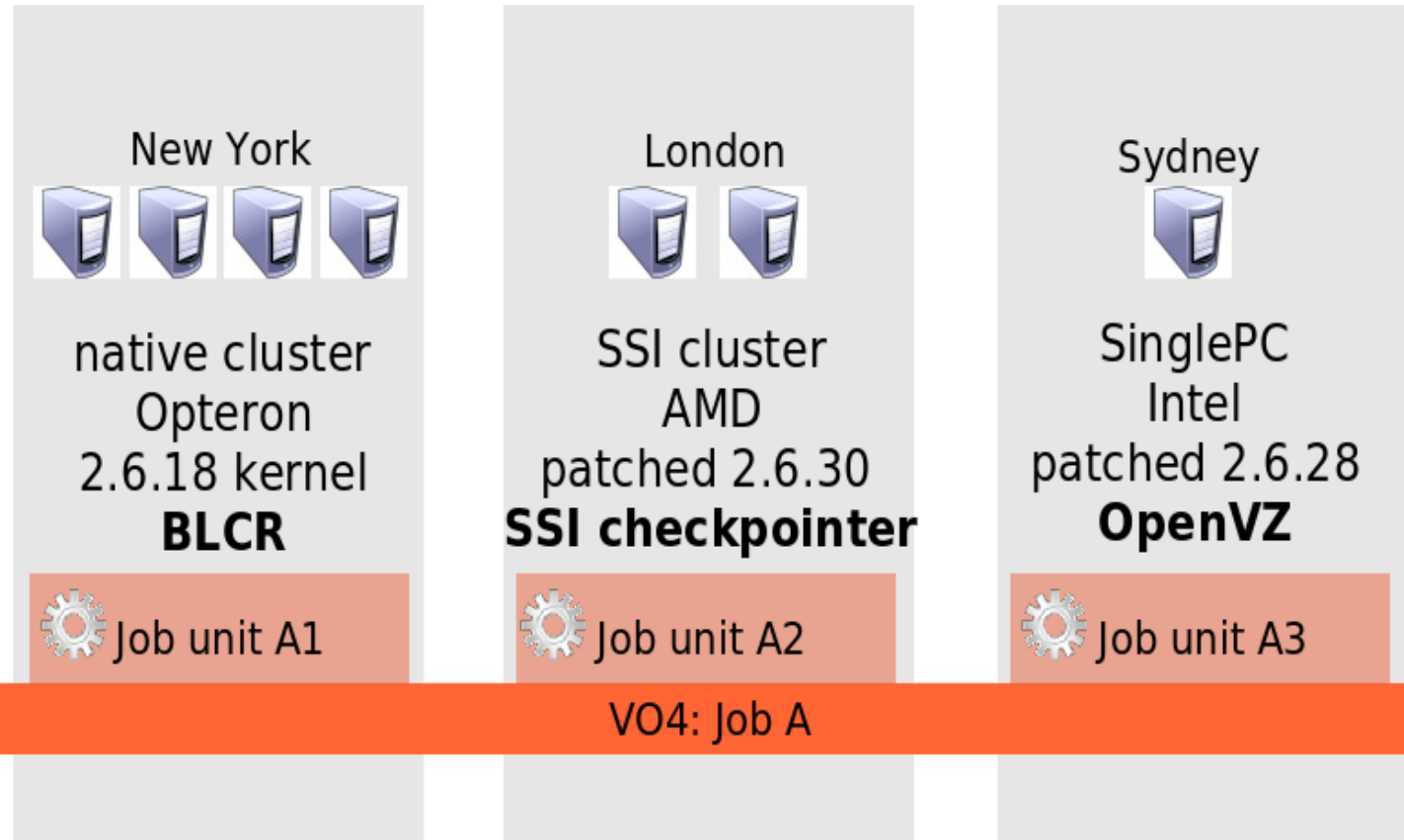
# What's the relation between existing checkpointers and grid checkpointing?

- **Existing Checkpointers do not deal with grid-specific aspects:**
  - resource dynamicity
  - adaptivity
  - security in the grid
  - ease-of-use
  - heterogeneity and interoperability





## Heterogeneity and Interoperability

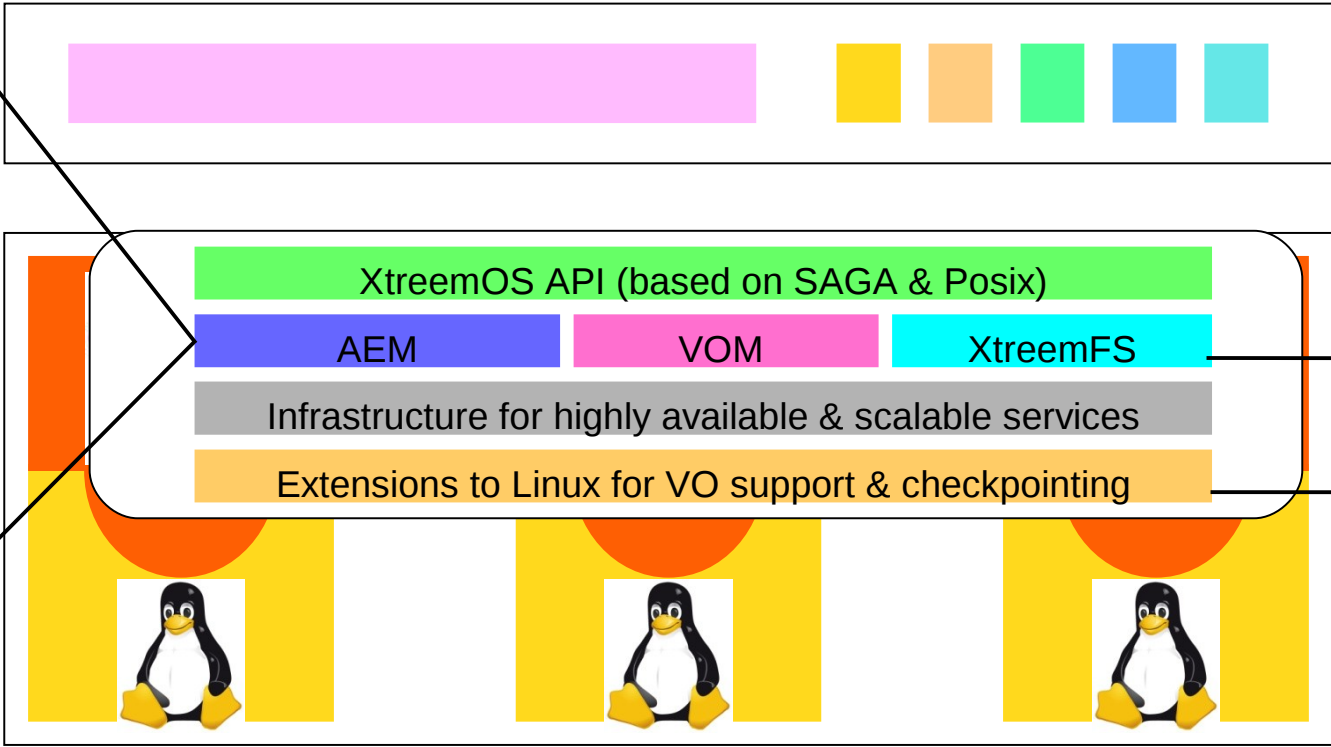


- **Solution: grid layer on top of existing checkpointers**



## AEM and grid checkpointing integration

- jsdl file  
fault tolerance exts.
- Job Mng  
fault tolerance exts.
- Execution Mng  
fault tolerance exts.



checkpoint files  
Common checkpointer API





Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux





Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux



Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux





Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux





Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux





Grid level

**Job Checkpointer**  
Job Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

**Job-unit Checkpointer**  
Execution Manager extension

Node Level

**Common Kernel Checkpointer API**

**Translation Library**

**Translation Library**

**LinuxSSI Kernel Checkp.**

**Linux Kernel Checkp.**



LinuxSSI cluster



Linux





# Coordinated Checkpointing Workflow



Job Checkpointer

Job-unit Checkpointer

Translation Library

LinuxSSI Checkp.



LinuxSSI cluster

Job-unit Checkpointer

Translation Library

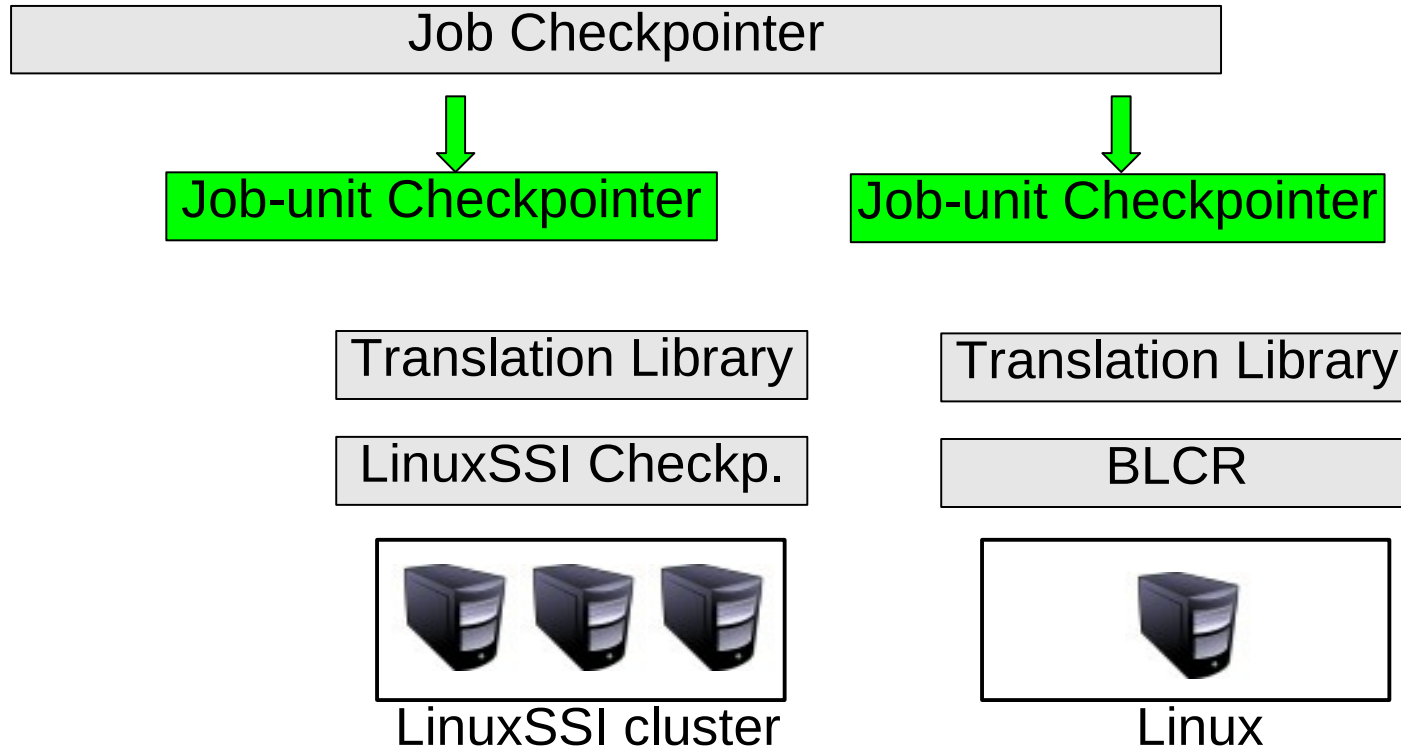
BLCR



Linux

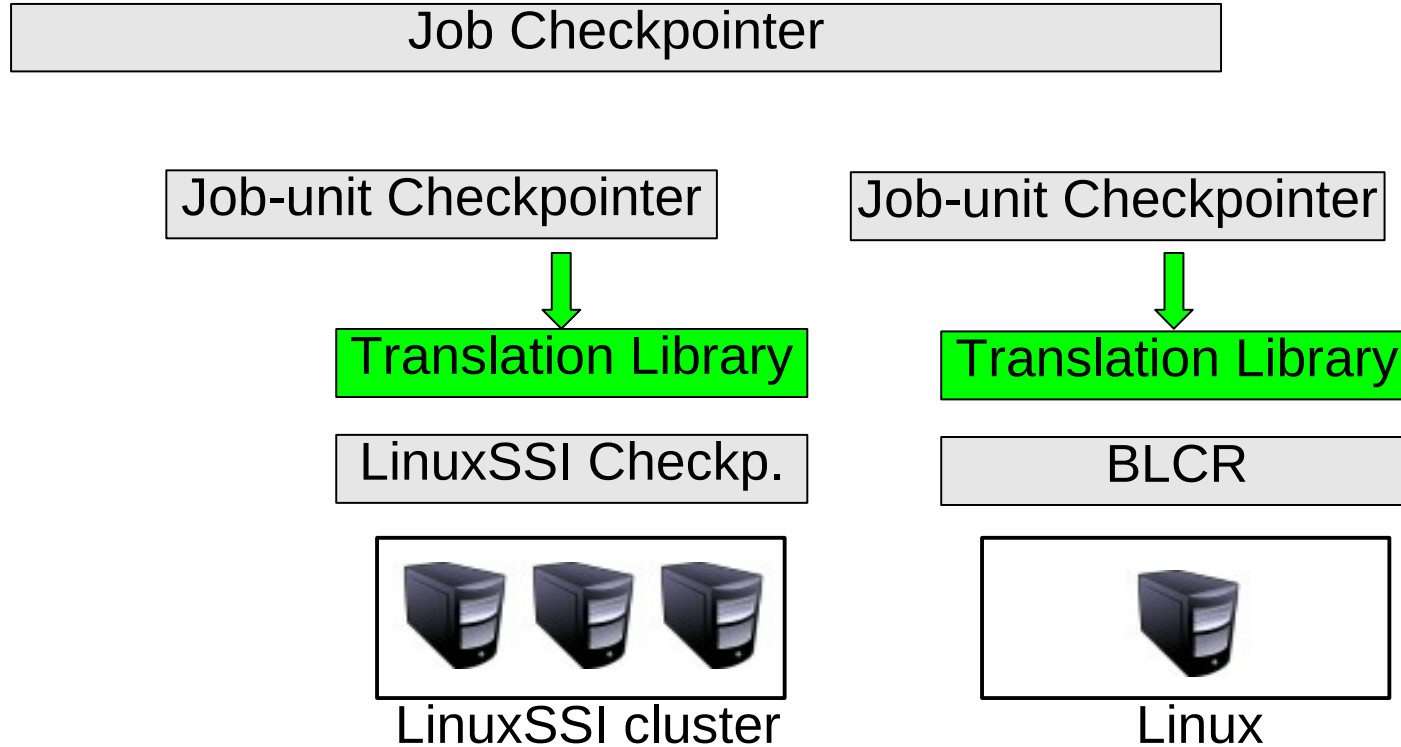


# Coordinated Checkpointing Workflow





# Coordinated Checkpointing Workflow





# Coordinated Checkpointing Workflow

Job Checkpointer

Job-unit Checkpointer

Translation Library



LinuxSSI Checkp.



LinuxSSI cluster

Job-unit Checkpointer

Translation Library



BLCR



Linux

job meta-data  
job-unit meta-data  
checkpointer images





# Independent Checkpointing Workflow

Job Checkpointer

→ Job-unit Checkpointer

Job-unit Checkpointer

Translation Library

Translation Library

LinuxSSI Checkp.

BLCR



LinuxSSI cluster

Linux



# Independent Checkpointing Workflow

Job Checkpointer

Job-unit Checkpointer

Job-unit Checkpointer



Translation Library

Translation Library

LinuxSSI Checkp.

BLCR



LinuxSSI cluster

Linux



# Independent Checkpointing Workflow

Job Checkpointer

Job-unit Checkpointer

Job-unit Checkpointer

Translation Library

Translation Library



LinuxSSI Checkp.

BLCR



LinuxSSI cluster

Linux

job meta-data  
job-unit meta-data  
checkpointer images



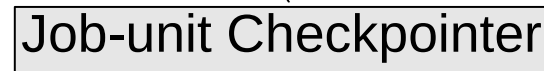
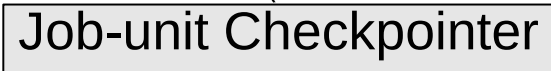


# Independent Restart Workflow (during application runtime)



receive determinants  
(create dependency graph)

wrappers for  
send, recv, etc.  
(LD\_PRELOAD)



LinuxSSI cluster



Linux

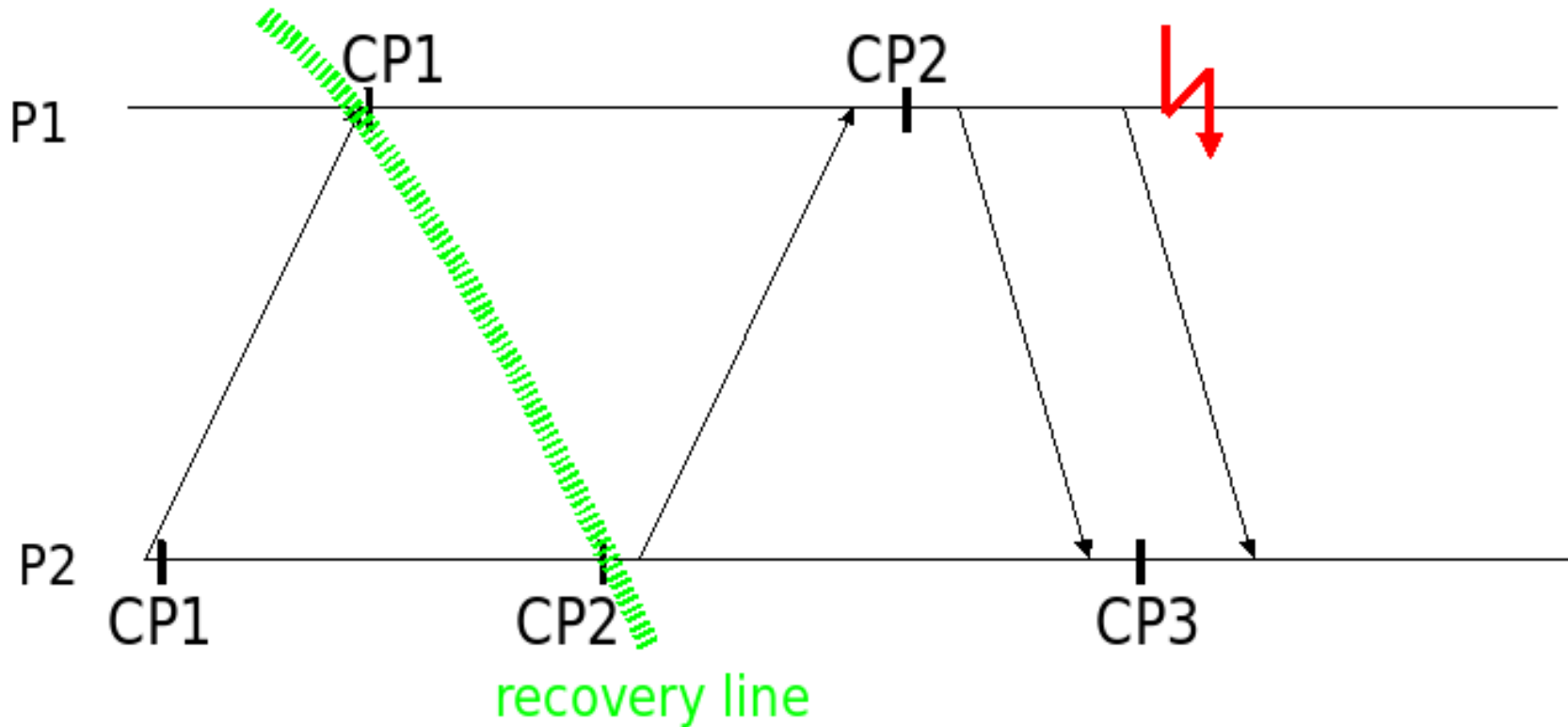




# Independent Restart Workflow

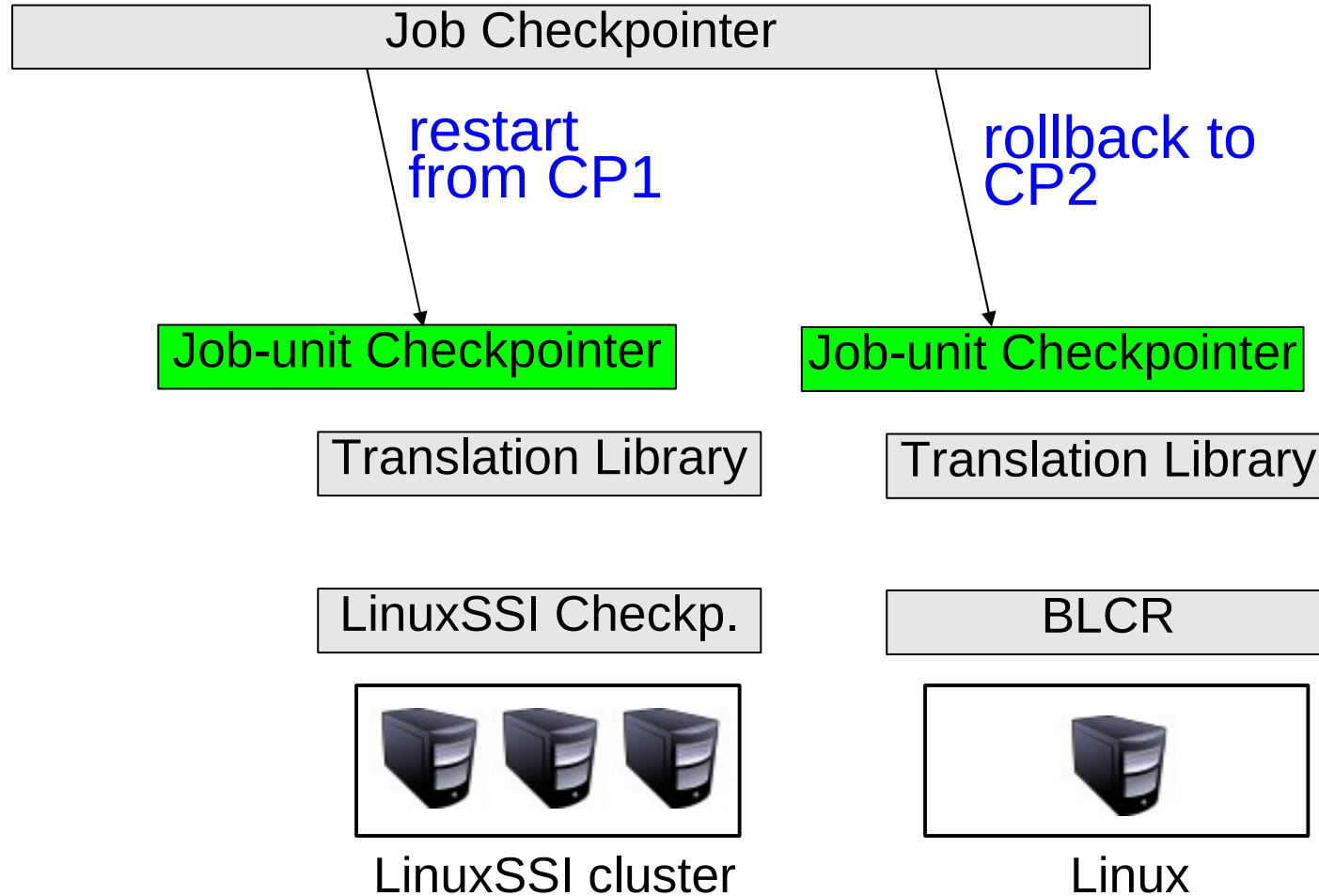
## Job Checkpointer

calculate recovery line from received determinants





# Independent Restart Workflow





- **Application-checkpointer matching**
  - jsdl extension planned
- **Translate individual calling semantics**
- **Translate job-to-Linux process group**
- **Resolve Checkpoint file dependencies**
- **Translate Callback Management semantics**
- **Security**
- **Checkpoint protocols and strategies**



- **To which extent must existing checkpointers be adapted to support various checkpointing protocols?**
  - global orchestration
- **Checkpoint/restart sequence splitting - checkpointing/restart stages**
  - stop
  - checkpoint
  - resume\_cp
  - rebuild
  - resume\_rst



## Checkpoint/Restart Callbacks

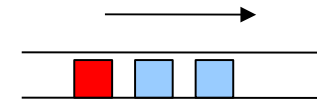
- **Pre-checkpoint, post-checkpoint, restart**
- **Integrate application-inherent knowledge**
- **Complement checkpointer incapacibilities**
- **Generic platform**
- **Common API for callback registration**





- **Coordinated and communication-induced cp**
- **Heterogenous checkpointers MUST cooperate with each other to handle in-transit messages**
- **Main Challenges:**
  - no checkpointer modifications
  - application transparency
  - migration support

BLCR



? OpenVZ

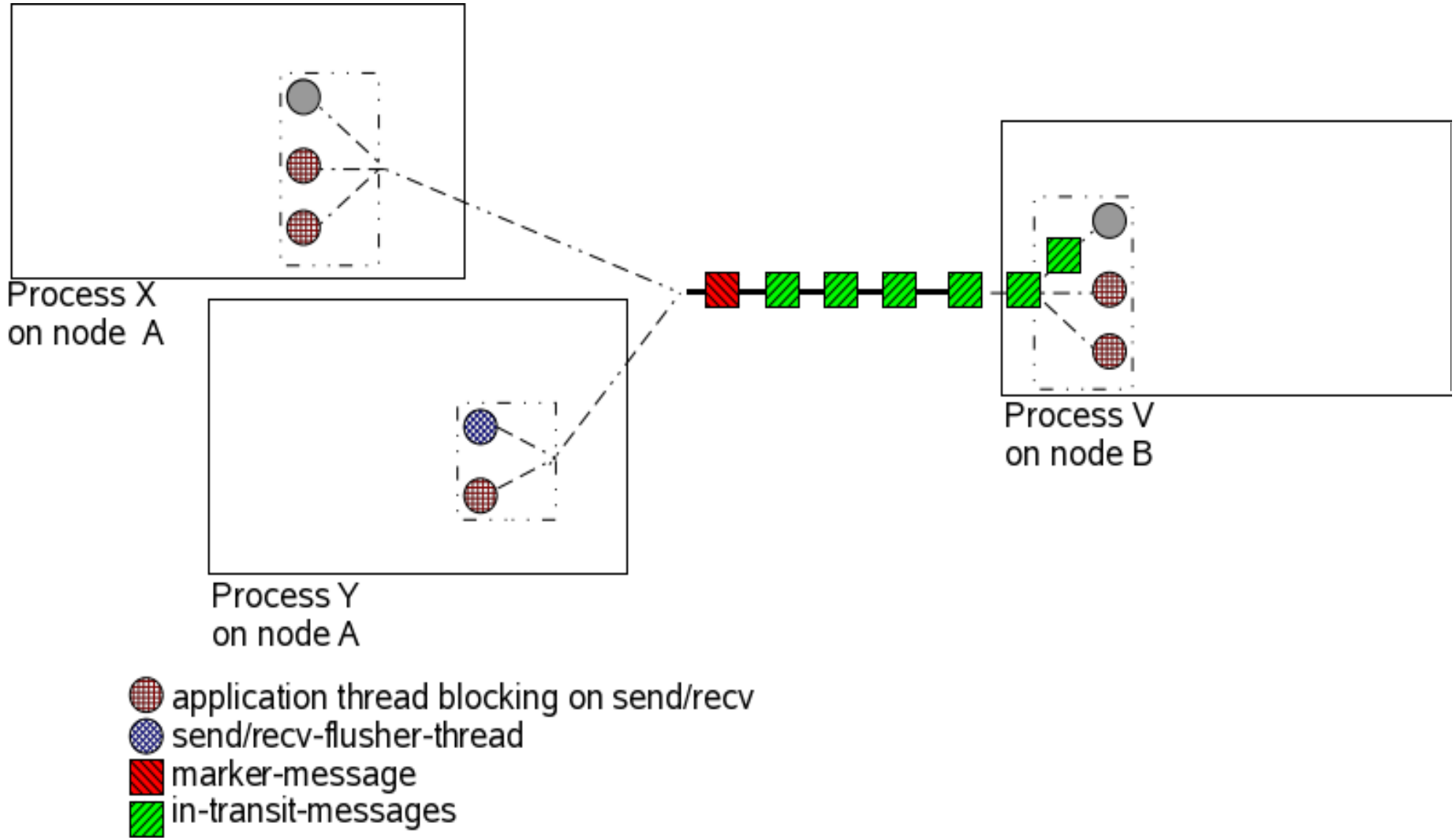




- **Coordinated and communication-induced cp**
- **Heterogenous checkpointers MUST cooperate with each other to handle in-transit messages**
- **Main Challenges:**
  - no checkpointer modifications
  - application transparency
  - migration support
- **Solution:**
  - channel flushing protocol



## Communication Channels 2/3





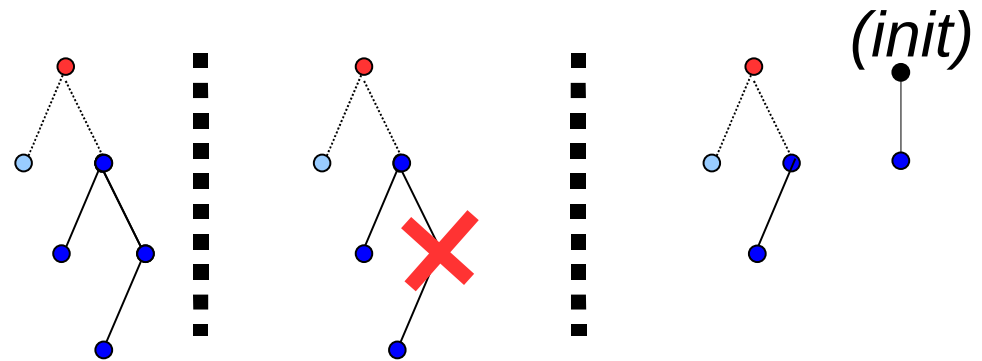
### ▪ Features:

- process states not sent across channel but on grid fs
- application transparent
- **support heterogeneous checkpointers**
- support concept of shared socket and socket descriptor in Linux





- Checkpointer input: process group ID, no process list
- Challenge: precise job-process-referencing
  - UNIX session ID
  - UNIX process group ID
  - root of process tree
  - LinuxSSI application id
  - OpenVZ containerID
  - cgroup
  - ...



“Checkpointing process groups in a Grid environment”, PDCAT08, Dunedin, NZ



### ▪ **Nodes:**

- Intel Core Duo
- 2GB RAM
- 3.0 GHz

### ▪ **Network:**

- Gigabit Ethernet

### ▪ **Application:**

- 5MB RAM, single process, single-threaded



# Preliminary Measurements

<b>Checkpoint (in ms)</b>					
	prepare	stop	checkpoint	resume	total
LinuxSSI (v0.9.3)	495,8	13,9	69,6	11,0	590,3
BLCR (v0.8.0)	381,2	40,1	250,5	5,3	677,1

<b>Restart (in ms)</b>					
			rebuild	resume	total
LinuxSSI (v0.9.3)			2597,7	12,6	2610,3
BLCR (v0.8.0)			1659,3	5,7	1665,0



- **Grid checkpointing integrated into XtreemOS**
  - heterogeneous checkpointers
  - main checkpointing protocols/strategies
  - open source
- **Adaptive checkpointing**
  - monitoring
  - further checkpoint protocolsstrategies
- **Grid5000**
- **XtreemOS Release 2.0**

	coordinated	incoordinated	incremental
BLCR	X	X	soon
LinuxSSI	X	X	X
OpenVZ	X	X	
MTCP	X	X	